Equivalence Properties by Typing in Cryptographic Branching Protocols

Véronique Cortier[1], Niklas Grimm[2], Joseph Lallemand[1], and Matteo Maffei[2]

[1] Université de Lorraine, CNRS, Inria, LORIA, France
[2] TU Wien, Austria

Abstract. Recently, many tools have been proposed for automatically analysing, in symbolic models, equivalence of security protocols. Equivalence is a property needed to state privacy properties or game-based properties like strong secrecy. Tools for a bounded number of sessions can decide equivalence but typically suffer from efficiency issues. Tools for an unbounded number of sessions like Tamarin or ProVerif prove a stronger notion of equivalence (diff-equivalence) that does not properly handle protocols with else branches.
Building upon a recent approach, we propose a type system for reasoning about branching protocols and dynamic keys. We prove our type system to entail equivalence, for all the standard primitives. Our type system has been implemented and shows a significant speedup compared to the tools for a bounded number of sessions, and compares similarly to ProVerif for an unbounded number of sessions. Moreover, we can also prove security of protocols that require a mix of bounded and unbounded number of sessions, which ProVerif cannot properly handle.

1 Introduction

Formal methods provide a rigorous and convenient framework for analysing security protocols. In particular, mature push-button analysis tools have emerged and have been successfully applied to many protocols from the literature in the context of *trace properties* such as authentication or confidentiality. These tools employ a variety of analysis techniques, such as model checking (e.g., Avispa [6] and Scyther [30]), Horn clause resolution (e.g., ProVerif [13]), term rewriting (e.g., Scyther [30] and Tamarin [37]), and type systems [36,19,20,21,12,35,16,33,17,7,18].

In the recent years, attention has been given also to equivalence properties, which are crucial to model privacy properties such as vote privacy [8,32], unlikability [5], or anonymity [9]. For example, consider an authentication protocol P_{pass} embedded in a biometric passport. P_{pass} preserves anonymity of passport holders if an attacker cannot distinguish an execution with Alice from an execution with Bob. This can be expressed by the equivalence $P_{pass}(Alice) \approx_t P_{pass}(Bob)$. Equivalence is also used to express properties closer to cryptographic games like strong secrecy.

Two main classes of tools have been developed for equivalence. First, in the case of an unbounded number of sessions (when the protocol is executed arbitrarily many times), equivalence is undecidable. Instead, the tools ProVerif [13,15] and Tamarin [37,11] try to prove a stronger property, namely diff-equivalence, that may be too strong e.g. in the context of voting. Tamarin covers a larger class of protocols but may require

some guidance from the user. Maude-NPA [34,39] also proves diff-equivalence but may have non-termination issues. Another class of tools aim at deciding equivalence, for bounded number of sessions. This is the case in particular of SPEC [31], APTE [23], Akiss [22], and SatEquiv [26]. SPEC, APTE, and Akiss suffer from efficiency issues and can typically not handle more than 3-4 sessions. SatEquiv is much more efficient but is limited to symmetric encryption and requires protocols to be well-typed, which often assumes some additional tagging of the protocol.

Our contribution. Following the approach of [28], we propose a novel technique for proving equivalence properties for a bounded number of sessions as well as an unbounded number of sessions (or a mix of both), based on typing. [28] proposes a first type system that entails trace equivalence $P \approx_t Q$, provided protocols use fixed (long-term) keys, identical in P and Q. In this paper, we target a larger class of protocols, that includes in particular key-exchange protocols and protocols whose security relies on branching on the secret. This is the case e.g. of the private authentication protocol [3], where agent B returns a true answer to A, encrypted with A's public key if A is one of his friends, and sends a decoy message (encrypted with a dummy key) otherwise.

We devise a new type system for reasoning about keys. In particular, we introduce bikeys to cover behaviours where keys in P differ from the keys in Q. We design new typing rules to reason about protocols that may branch differently (in P and Q), depending on the input. Following the approach of [28], our type system collects sent messages into constraints that are required to be consistent. Intuitively, the type system guarantees that any execution of P can be matched by an execution of Q, while consistency imposes that the resulting sequences of messages are indistinguishable for an attacker. We had to entirely revisit the approach of [28] and prove a finer invariant in order to cope with the case where keys are used as variables. Specifically, most of the rules for encryption, signature, and decryption had to be adapted to accommodate the flexible usage of keys. For messages, we had to modify the rules for keys and encryption, in order to encrypt messages with keys of different type (bi-key type), instead of only fixed keys. We show that our type system entails equivalence for the standard notion of trace equivalence [24] and we devise a procedure for proving consistency. This yields an efficient approach for *proving* equivalence of protocols for a bounded and an unbounded number of sessions (or a combination of both).

We implemented a prototype of our type-checker that we evaluate on a set of examples, that includes private authentication, the BAC protocol (of the biometric passport), as well as Helios together with the setup phase. Our tool requires a light type annotation that specifies which keys and names are likely to be secret or public and the form of the messages encrypted by a given key. This can be easily inferred from the structure of the protocol. Our type-checker outperforms even the most efficient existing tools for a bounded number of sessions by two (for examples with few processes) to three (for examples with more processes) orders of magnitude. Note however that these tools *decide* equivalence while our type system is incomplete. In the case of an unbounded number of sessions, on our examples, the performance is comparable to ProVerif, one of the most popular tools. We consider in particular vote privacy in the Helios protocol, in the case of a dishonest ballot board, with no revote (as the protocol is insecure otherwise). ProVerif fails to handle this case as it cannot (faithfully) consider a mix of bounded and

unbounded number of sessions. Compared to [28], our analysis includes the setup phase (where voters receive the election key), which could not be considered before.

2 High-level description

2.1 Background

Trace equivalence of two processes is a property that guarantees that an attacker observing the execution of either of the two processes cannot decide which one it is. Previous work [28] has shown how trace equivalence can be proved statically using a type system combined with a constraint checking procedure. The type system consists of typing rules of the form $\Gamma \vdash P \sim Q \to C$, meaning that in an environment Γ two processes P and Q are equivalent if the produced set of constraints C, encoding the attacker observables, is consistent.

The typing environment Γ is a mapping from nonces, keys, and variables to types. Nonces are assigned security labels with a confidentiality and an integrity component, e.g. `HL` for high confidentiality and low integrity. Key types are of the form $\mathrm{key}^l(T)$ where l is the security label of the key and T is the type of the payload. Key types are crucial to convey typing information from one process to another one. Normally, we cannot make any assumptions about values received from the network – they might possibly originate from the attacker. If we however successfully decrypt a message using a secret symmetric key, we know that the result is of the key's payload type. This is enforced on the sender side, whenever outputting an encryption.

A core assumption of virtually any efficient static analysis for equivalence is uniform execution, meaning that the two processes of interest always take the same branch in a branching statement. For instance, this means that all decryptions must always succeed or fail equally in the two processes. For this reason, previous work introduced a restriction to allow only encryption and decryption with keys whose equality could be statically proved.

2.2 Limitation

There are however protocols that require non-uniform execution for a proof of trace equivalence, e.g., the private authentication protocol [3]. The protocol aims at authenticating B to A, anonymously w.r.t. other agents. More specifically, agent B may refuse to communicate with agent A but a third agent D should not learn whether B declines communication with A or not. The protocol can be informally described as follows, where $\mathtt{pk}(k)$ denotes the public key associated to key k, and $\mathtt{aenc}(M, \mathtt{pk}(k))$ denotes the asymmetric encryption of message M with this public key.

$$
\begin{array}{ll}
A \to B: & \mathtt{aenc}(\langle N_a, \mathtt{pk}(k_a)\rangle, \mathtt{pk}(k_b)) \\
B \to A: & \begin{cases} \mathtt{aenc}(\langle N_a, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) & \text{if } B \text{ accepts } A\text{'s request} \\ \mathtt{aenc}(N_b, \mathtt{pk}(k)) & \text{if } B \text{ declines } A\text{'s request} \end{cases}
\end{array}
$$

If B declines to communicate with A, he sends a decoy message $\mathtt{aenc}(N_b, \mathtt{pk}(k))$ where $\mathtt{pk}(k)$ is a decoy key (no one knows the private key k).

$$
\begin{array}{rll}
\Gamma(k_b, k_b) & = \text{key}^{\mathtt{HH}}(\mathtt{HL} * \mathtt{LL}) & \text{initial message uses same key on both sides} \\
\Gamma(k_a, k) & = \text{key}^{\mathtt{HH}}(\mathtt{HL}) & \text{authentication succeeded on the left, failed on the right} \\
\Gamma(k, k_c) & = \text{key}^{\mathtt{HH}}(\mathtt{HL}) & \text{authentication succeeded on the right, failed on the left} \\
\Gamma(k_a, k_c) & = \text{key}^{\mathtt{HH}}(\mathtt{HL}) & \text{authentication succeeded on both sides} \\
\Gamma(k, k) & = \text{key}^{\mathtt{HH}}(\mathtt{HL}) & \text{authentication failed on both sides}
\end{array}
$$

Fig. 1. Key types for the private authentication protocol

2.3 Encrypting with different keys

Let $P_a(k_a, \mathtt{pk}(k_b))$ model agent A willing to talk with B, and $P_b(k_b, \mathtt{pk}(k_a))$ model agent B willing to talk with A (and declining requests from other agents). We model the protocol as:

$$
\begin{array}{l}
P_a(k_a, pk_b) = \mathtt{new}\ N_a.\mathtt{out}(\mathtt{aenc}(\langle N_a, \mathtt{pk}(k_a)\rangle, pk_b)).\ \mathtt{in}(z) \\
P_b(k_b, pk_a) = \mathtt{new}\ N_b.\ \mathtt{in}(x). \\
\qquad\qquad \mathtt{let}\ y = \mathtt{adec}(x, k_b)\ \mathtt{in}\ \mathtt{let}\ y_1 = \pi_1(y)\ \mathtt{in}\ \mathtt{let}\ y_2 = \pi_2(y)\ \mathtt{in} \\
\qquad\qquad\quad \mathtt{if}\ y_2 = pk_a\ \mathtt{then} \\
\qquad\qquad\qquad \mathtt{out}(\mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, pk_a)) \\
\qquad\qquad\quad \mathtt{else}\ \mathtt{out}(\mathtt{aenc}(N_b, \mathtt{pk}(k)))
\end{array}
$$

where $\mathtt{adec}(M, k)$ denotes asymmetric decryption of message M with private key k. We model anonymity as the following equivalence, intuitively stating that an attacker should not be able to tell whether B accepts requests from the agent A or C:

$$
P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_a)) \approx_t P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_c))
$$

We now show how we can type the protocol in order to show trace equivalence. The initiator P_a is trivially executing uniformly, since it does not contain any branching operations. We hence focus on typing the responder P_b.

The beginning of the responder protocol can be typed using standard techniques. Then however, we perform the test $y_2 = \mathtt{pk}(k_a)$ on the left side and $y_2 = \mathtt{pk}(k_c)$ on the right side. Since we cannot statically determine the result of the two equality checks – and thus guarantee uniform execution – we have to typecheck the four possible combinations of `then` and `else` branches. This means we have to typecheck outputs of encryptions that use different keys on the left and the right side.

To deal with this we do not assign types to single keys, but rather to pairs of keys (k, k') – which we call *bikeys* – where k is the key used in the left process and k' is the key used in the right process. The key types used for typing are presented in Fig. 1.

As an example, we consider the combination of the `then` branch on the left with the `else` branch on the right. This combination occurs when A is successfully authenticated on the left side, while being rejected on the right side. We then have to typecheck B's positive answer together with the decoy message: $\Gamma \vdash \mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k)) : \mathtt{LL}$. For this we need the type for the bikey (k_a, k).

2.4 Decrypting non-uniformly

When decrypting a ciphertext that was potentially generated using two different keys on the left and the right side, we have to take all possibilities into account. Consider the following extension of the process P_a where agent A decrypts B's message.

$$\begin{aligned} P_a(k_a, pk_b) = \mathtt{new}\ N_a.\mathtt{out}(\mathtt{aenc}(\langle N_a, \mathtt{pk}(k_a)\rangle, pk_b)).\ \mathtt{in}(z).\\ \mathtt{let}\ z' = \mathtt{adec}(z, k_a)\ \mathtt{in}\ \mathtt{out}(1)\\ \mathtt{else}\ \mathtt{out}(0) \end{aligned}$$

In the decryption, there are the following possible cases:

- The message is a valid encryption supplied by the attacker (using the public key $\mathtt{pk}(k_a)$), so we check the `then` branch on both sides with $\Gamma(z') = \mathtt{LL}$.
- The message is not a valid encryption supplied by the attacker so we check the `else` branch on both sides.
- The message is a valid response from B. The keys used on the left and the right are then one of the four possible combinations $(k_a, k), (k_a, k_c), (k, k_c)$ and (k, k).
  - In the first two cases the decryption will succeed on the left and fail on the right. We hence check the `then` branch on the left with $\Gamma(z') = \mathtt{HL}$ with the `else` branch on the right. If the type $\Gamma(k_a, k)$ were different from $\Gamma(k_a, k_c)$, we would check this combination twice, using the two different payload types.
  - In the remaining two cases the decryption will fail on both sides. We hence would have to check the two `else` branches (which however we already did).

While checking the `then` branch together with the `else` branch, we have to check $\Gamma \vdash 1 \sim 0 : \mathtt{LL}$, which rightly fails, as the protocol does not guarantee trace equivalence.

3 Model

In symbolic models, security protocols are typically modelled as processes of a process algebra, such as the applied pi-calculus [2]. We present here a calculus used in [28] and inspired from the calculus underlying the ProVerif tool [14]. This section is mostly an excerpt of [28], recalled here for the sake of completeness, and illustrated with the private authentication protocol.

3.1 Terms

Messages are modelled as terms. We assume an infinite set of names $\mathcal{N}$ for nonces, further partitioned into the set $\mathcal{FN}$ of free nonces (created by the attacker) and the set $\mathcal{BN}$ of bound nonces (created by the protocol parties), an infinite set of names $\mathcal{K}$ for keys similarly split into $\mathcal{FK}$ and $\mathcal{BK}$, and an infinite set of variables $\mathcal{V}$. Cryptographic primitives are modelled through a *signature* $\mathcal{F}$, that is, a set of function symbols, given with their arity (*i.e.* the number of arguments). Here, we consider the following signature:

$$\mathcal{F}_c = \{\mathtt{pk}, \mathtt{vk}, \mathtt{enc}, \mathtt{aenc}, \mathtt{sign}, \langle\cdot,\cdot\rangle, \mathtt{h}\}$$

that models respectively public and verification key, symmetric and asymmetric encryption, concatenation and hash. The companion primitives (symmetric and asymmetric decryption, signature check, and projections) are represented by the following signature:

$$\mathcal{F}_d = \{\mathtt{dec}, \mathtt{adec}, \mathtt{checksign}, \pi_1, \pi_2\}$$

We also consider a set $\mathcal{C}$ of (public) constants (used as agent names for instance). Given a signature $\mathcal{F}$, a set of names $\mathcal{N}$, and a set of variables $\mathcal{V}$, the set of *terms* $\mathcal{T}(\mathcal{F}, \mathcal{V}, \mathcal{N})$ is the set inductively defined by applying functions to variables in $\mathcal{V}$ and names in $\mathcal{N}$. We denote by $\mathrm{names}(t)$ (resp. $\mathrm{vars}(t)$) the set of names (resp. variables) occurring in t. A term is *ground* if it does not contain variables.

We consider the set $\mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d \cup \mathcal{C}, \mathcal{V}, \mathcal{N} \cup \mathcal{K})$ of *cryptographic terms*, simply called *terms*. *Messages* are terms with constructors from $\mathcal{T}(\mathcal{F}_c \cup \mathcal{C}, \mathcal{V}, \mathcal{N} \cup \mathcal{K})$. We assume the set of variables to be split into two subsets $\mathcal{V} = \mathcal{X} \uplus \mathcal{AX}$ where $\mathcal{X}$ are variables used in processes while $\mathcal{AX}$ are variables used to store messages. An *attacker term* is a term from $\mathcal{T}(\mathcal{F}_c \cup \mathcal{F}_d \cup \mathcal{C}, \mathcal{AX}, \mathcal{FN} \cup \mathcal{FK})$. In particular, an attacker term cannot use nonces and keys created by the protocol's parties.

A *substitution* $\sigma = \{M_1/x_1, \ldots, M_k/x_k\}$ is a mapping from variables $x_1, \ldots, x_k \in \mathcal{V}$ to messages $M_1, \ldots, M_k$. We let $\mathrm{dom}(\sigma) = \{x_1, \ldots, x_k\}$. We say that σ is ground if all messages $M_1, \ldots, M_k$ are ground. We let $\mathrm{names}(\sigma) = \bigcup_{1 \le i \le k} \mathrm{names}(M_i)$. The application of a substitution σ to a term t is denoted $t\sigma$ and is defined as usual.

The *evaluation* of a term t, denoted $t\downarrow$, corresponds to the bottom-up application of the cryptographic primitives and is recursively defined as follows.

$$\begin{array}{rl}
u\downarrow = u & \text{if } u \in \mathcal{N} \cup \mathcal{V} \cup \mathcal{K} \cup \mathcal{C} \\
\mathtt{pk}(t)\downarrow = \mathtt{pk}(t\downarrow) & \text{if } t\downarrow \in \mathcal{K} \\
\mathtt{vk}(t)\downarrow = \mathtt{vk}(t\downarrow) & \text{if } t\downarrow \in \mathcal{K} \\
\mathtt{h}(t)\downarrow = \mathtt{h}(t\downarrow) & \text{if } t\downarrow \neq \bot \\
\langle t_1, t_2\rangle\downarrow = \langle t_1\downarrow, t_2\downarrow\rangle & \text{if } t_1\downarrow \neq \bot \text{ and } t_2\downarrow \neq \bot \\
\mathtt{enc}(t_1, t_2)\downarrow = \mathtt{enc}(t_1\downarrow, t_2\downarrow) & \text{if } t_1\downarrow \neq \bot \text{ and } t_2\downarrow \in \mathcal{K} \\
\mathtt{sign}(t_1, t_2)\downarrow = \mathtt{sign}(t_1\downarrow, t_2\downarrow) & \text{if } t_1\downarrow \neq \bot \text{ and } t_2\downarrow \in \mathcal{K} \\
\mathtt{aenc}(t_1, t_2)\downarrow = \mathtt{aenc}(t_1\downarrow, t_2\downarrow) & \text{if } t_1\downarrow \neq \bot \text{ and } t_2\downarrow = \mathtt{pk}(k) \\
\multicolumn{2}{c}{\text{for some } k \in \mathcal{K}}
\end{array}$$

$$\begin{array}{rl}
\pi_1(t)\downarrow = t_1 & \text{if } t\downarrow = \langle t_1, t_2\rangle \\
\pi_2(t)\downarrow = t_2 & \text{if } t\downarrow = \langle t_1, t_2\rangle \\
\mathtt{dec}(t_1, t_2)\downarrow = t_3 & \text{if } t_1\downarrow = \mathtt{enc}(t_3, t_4) \text{ and } t_4 = t_2\downarrow \\
\mathtt{adec}(t_1, t_2)\downarrow = t_3 & \text{if } t_1\downarrow = \mathtt{aenc}(t_3, \mathtt{pk}(t_4)) \text{ and } t_4 = t_2\downarrow \\
\mathtt{checksign}(t_1, t_2)\downarrow = t_3 & \text{if } t_1\downarrow = \mathtt{sign}(t_3, t_4) \text{ and } t_2\downarrow = \mathtt{vk}(t_4) \\
t\downarrow = \bot & \text{otherwise}
\end{array}$$

Note that the evaluation of term t succeeds only if the underlying keys are atomic and always returns a message or $\bot$. For example we have $\pi_1(\langle a, b\rangle)\downarrow = a$, while $\mathtt{dec}(\mathtt{enc}(a, \langle b, b\rangle), \langle b, b\rangle)\downarrow = \bot$, because the key is non atomic. We write $t =_\downarrow t'$ if $t\downarrow = t'\downarrow$.

Destructors used in processes:

$$d ::= \mathtt{dec}(x,t) \mid \mathtt{adec}(x,t) \mid \mathtt{checksign}(x,t') \mid \pi_1(x) \mid \pi_2(x)$$

where $x \in \mathcal{X}$, $t \in \mathcal{K} \cup \mathcal{X}$, $t' \in \{\mathtt{vk}(k) | k \in \mathcal{K}\} \cup \mathcal{X}$.

Processes:

$$\begin{aligned} P, Q ::= {} & \mathtt{0} \mid \mathtt{new}\ n.P \mid \mathtt{out}(M).P \mid \mathtt{in}(x).P \mid (P \mid Q) \mid !P \\ & \mid \mathtt{let}\ x = d\ \mathtt{in}\ P\ \mathtt{else}\ Q \mid \mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q \end{aligned}$$

where $n \in \mathcal{BN} \cup \mathcal{BK}$, $x \in \mathcal{X}$, and M, N are messages.

Fig. 2. Syntax for processes.

3.2 Processes

Security protocols describe how messages should be exchanged between participants. We model them through a process algebra, whose syntax is displayed in Fig. 2. We identify processes up to α-renaming, *i.e.*, avoiding substitution of bound names and variables, which are defined as usual. Furthermore, we assume that all bound names, keys, and variables in the process are distinct.

A *configuration* of the system is a tuple $(\mathcal{P}; \phi; \sigma)$ where:

- $\mathcal{P}$ is a multiset of processes that represents the current active processes;
- ϕ is a substitution with $\mathrm{dom}(\phi) \subseteq \mathcal{AX}$ and for any $x \in \mathrm{dom}(\phi)$, $\phi(x)$ (also denoted $x\phi$) is a message that only contains variables in $\mathrm{dom}(\sigma)$. ϕ represents the terms that have been sent;
- σ is a ground substitution.

The semantics of processes is given through a transition relation $\xrightarrow{\alpha}$, defined in Figure 3 (τ denotes a silent action). The relation $\xrightarrow{w}_*$ is defined as the reflexive transitive closure of $\xrightarrow{\alpha}$, where w is the concatenation of all actions. We also write equality up to silent actions $=_\tau$.

Intuitively, process $\mathtt{new}\ n.P$ creates a fresh nonce or key, and behaves like P. Process $\mathtt{out}(M).P$ emits M and behaves like P, provided that the evaluation of M is successful. The corresponding message is stored in the frame ϕ, corresponding to the attacker knowledge. A process may input any message that an attacker can forge (rule IN) from her knowledge ϕ, using a recipe R to compute a new message from ϕ. Note that all names are initially assumed to be secret. Process $P \mid Q$ corresponds to the parallel composition of P and Q. Process $\mathtt{let}\ x = d\ \mathtt{in}\ P\ \mathtt{else}\ Q$ behaves like P in which x is replaced by d if d can be successfully evaluated and behaves like Q otherwise. Process $\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q$ behaves like P if M and N correspond to two equal messages and behaves like Q otherwise. The replicated process $!P$ behaves as an unbounded number of copies of P.

A *trace* of a process P is any possible sequence of transitions in the presence of an attacker that may read, forge, and send messages. Formally, the set of traces $\mathrm{trace}(P)$ is defined as follows.

$$\mathrm{trace}(P) = \{(w, \phi, \sigma) | (\{P\}; \emptyset; \emptyset) \xrightarrow{w}_* (\mathcal{P}; \phi; \sigma)\}$$

$$(\{P_1 \mid P_2\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P_1, P_2\} \cup \mathcal{P}; \phi; \sigma) \quad \text{PAR}$$

$$(\{\mathtt{0}\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\mathcal{P}; \phi; \sigma) \quad \text{ZERO}$$

$$(\{\mathtt{new}\ n.P\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P\} \cup \mathcal{P}; \phi; \sigma) \quad \text{NEW}$$

$$(\{\mathtt{new}\ k.P\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P\} \cup \mathcal{P}; \phi; \sigma) \quad \text{NEWKEY}$$

$$(\{\mathtt{out}(t).P\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\mathtt{new}\ ax_n.\mathtt{out}(ax_n)} (\{P\} \cup \mathcal{P}; \phi \cup \{t/ax_n\}; \sigma) \quad \text{OUT}$$
if $t\sigma$ is a ground term, $(t\sigma) \downarrow \neq \bot$, $ax_n \in \mathcal{AX}$ and $n = |\phi| + 1$

$$(\{\mathtt{in}(x).P\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\mathtt{in}(R)} (\{P\} \cup \mathcal{P}; \phi; \sigma \cup \{(R\phi\sigma) \downarrow /x\}) \quad \text{IN}$$
if R is an attacker term such that $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi)$, and $(R\phi\sigma) \downarrow \neq \bot$

$$(\{\mathtt{let}\ x = d\ \mathtt{in}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P\} \cup \mathcal{P}; \phi; \sigma \cup \{(d\sigma) \downarrow /x\}) \quad \text{LET-IN}$$
if $d\sigma$ is ground and $(d\sigma) \downarrow \neq \bot$

$$(\{\mathtt{let}\ x = d\ \mathtt{in}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{Q\} \cup \mathcal{P}; \phi; \sigma) \quad \text{LET-ELSE}$$
if $d\sigma$ is ground and $(d\sigma) \downarrow = \bot$, *i.e.* d fails

$$(\{\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P\} \cup \mathcal{P}; \phi; \sigma) \quad \text{IF-THEN}$$
if M, N are messages such that $M\sigma$, $N\sigma$ are ground, $(M\sigma) \downarrow \neq \bot$, $(N\sigma) \downarrow \neq \bot$, and $M\sigma = N\sigma$

$$(\{\mathtt{if}\ M = N\ \mathtt{then}\ P\ \mathtt{else}\ Q\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{Q\} \cup \mathcal{P}; \phi; \sigma) \quad \text{IF-ELSE}$$
if M, N are messages such that $M\sigma$, $N\sigma$ are ground and $(M\sigma) \downarrow = \bot$ or $(N\sigma) \downarrow = \bot$ or $M\sigma \neq N\sigma$

$$(\{!P\} \cup \mathcal{P}; \phi; \sigma) \xrightarrow{\tau} (\{P, !P\} \cup \mathcal{P}; \phi; \sigma) \quad \text{REPL}$$

Fig. 3. Semantics

Example 1. Consider the private authentication protocol (PA) presented in Section 2. The process $P_b(k_b, \mathtt{pk}(k_a))$ corresponding to responder B answering a request from A has already been defined in Section 2.3. The process $P_a(k_a, \mathtt{pk}(k_b))$ corresponding A willing to talk to B is:

$$P_a(k_a, pk_b) = \mathtt{new}\ N_a.\mathtt{out}(\mathtt{aenc}(\langle N_a, \mathtt{pk}(k_a)\rangle, pk_b)).\ \mathtt{in}(z)$$

Altogether, a session between A and B is represented by the process:

$$P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_a))$$

where $k_a, k_b \in \mathcal{BK}$, which models that the attacker initially does not know k_a, k_b.

An example of a trace describing an "honest" execution, where the attacker does not interfere with the intended run of the protocol, can be written as (tr, ϕ) where

$$tr =_\tau \mathtt{new}\ x_1.\mathtt{out}(x_1).\mathtt{in}(x_1).\mathtt{new}\ x_2.\mathtt{out}(x_2).\mathtt{in}(x_2)$$

and

$$\phi = \{x_1 \mapsto \mathtt{aenc}(\langle N_a, \mathtt{pk}(k_a)\rangle, \mathtt{pk}(k_b)), x_2 \mapsto \mathtt{aenc}(\langle N_a, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a))\}.$$

The trace tr describes A outputting the first message of the protocol, which is stored in $\phi(x_1)$. The attacker then simply forwards $\phi(x_1)$ to B. B then performs several silent actions (decrypting the message, comparing its content to $\mathtt{pk}(k_a)$), and outputs a response, which is stored in $\phi(x_2)$ and forwarded to A by the attacker.

$$
\begin{array}{lcl}
l & ::= & \mathtt{LL} \mid \mathtt{HL} \mid \mathtt{HH} \\
KT & ::= & \mathrm{key}^{l}(T) \mid \mathrm{eqkey}^{l}(T) \mid \mathrm{seskey}^{l,a}(T) \text{ with } a \in \{1, \infty\} \\
T & ::= & l \mid T * T \mid T \vee T \mid [\![\tau_n^{l,a} \,;\, \tau_m^{l',a}]\!] \text{ with } a \in \{1, \infty\} \\
 & & \mid KT \mid \mathrm{pkey}(KT) \mid \mathrm{vkey}(KT) \mid (T)_T \mid \{T\}_T
\end{array}
$$

Fig. 4. Types for terms

3.3 Equivalence

When processes evolve, sent messages are stored in a substitution ϕ while the values of variables are stored in σ. A *frame* is simply a substitution ψ where $\mathrm{dom}(\psi) \subseteq \mathcal{AX}$. It represents the knowledge of an attacker. In what follows, we will typically consider $\phi\sigma$.

Intuitively, two sequences of messages are indistinguishable to an attacker if he cannot perform any test that could distinguish them. This is typically modelled as static equivalence [2]. Here, we consider of variant of [2] where the attacker is also given the ability to observe when the evaluation of a term fails, as defined for example in [25].

Definition 1 (Static Equivalence). *Two ground frames ϕ and ϕ' are statically equivalent if and only if they have the same domain, and for all attacker terms R, S with variables in $\mathrm{dom}(\phi) = \mathrm{dom}(\phi')$, we have*

$$(R\phi =_{\downarrow} S\phi) \iff (R\phi' =_{\downarrow} S\phi')$$

Then two processes P and Q are in equivalence if no matter how the adversary interacts with P, a similar interaction may happen with Q, with equivalent resulting frames.

Definition 2 (Trace Equivalence). *Let P, Q be two processes. We write $P \sqsubseteq_t Q$ if for all $(s, \phi, \sigma) \in \mathrm{trace}(P)$, there exists $(s', \phi', \sigma') \in \mathrm{trace}(Q)$ such that $s =_\tau s'$ and $\phi\sigma$ and $\phi'\sigma'$ are statically equivalent. We say that P and Q are trace equivalent, and we write $P \approx_t Q$, if $P \sqsubseteq_t Q$ and $Q \sqsubseteq_t P$.*

Note that this definition already includes the attacker's behaviour, since processes may input any message forged by the attacker.

Example 2. As explained in Section 2, anonymity is modelled as an equivalence property. Intuitively, an attacker should not be able to know which agents are executing the protocol. In the case of protocol PA, presented in Example 1, the anonymity property can be modelled by the following equivalence:

$$P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_a)) \approx_t P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_c))$$

4 A type system for dynamic keys

Types In our type system we give types to pairs of messages – one from the left process and one from the right one. We store the types of nonces, variables, and keys in a typing environment Γ. While we store a type for a single nonce or variable occurring

$$\frac{}{\mathrm{eqkey}^l(T) <: \mathrm{key}^l(T)}\ (\textsc{SEqKey}) \qquad \frac{}{\mathrm{seskey}^{l,a}(T) <: \mathrm{eqkey}^l(T)}\ (\textsc{SSesKey})$$

$$\frac{}{\mathrm{key}^l(T) <: l}\ (\textsc{SKey}) \qquad \frac{T <: \mathrm{eqkey}^l(T')}{\mathrm{pkey}(T) <: \mathtt{LL}}\ (\textsc{SPubKey}) \qquad \frac{T <: \mathrm{eqkey}^l(T')}{\mathrm{vkey}(T) <: \mathtt{LL}}\ (\textsc{SVKey})$$

$$\frac{T <: T'}{(T)_{T''} <: (T')_{T''}}\ (\textsc{SEnc}) \qquad \frac{T <: T'}{\{T\}_{T''} <: \{T'\}_{T''}}\ (\textsc{SAEnc})$$

Fig. 5. Selected subtyping rules

in both processes, we assign a potentially different type to every different combination of keys (k, k') used in the left and right process – so called *bikeys*. This is an important non-standard feature that enables us to type protocols using different encryption and decryption keys.

The types for messages are defined in Fig. 4 and explained below. Selected subtyping rules are given in Fig. 5. We assume three security labels $\mathtt{HH}$, $\mathtt{HL}$ and $\mathtt{LL}$, ranged over by l, whose first (resp. second) component denotes the confidentiality (resp. integrity) level. Intuitively, values of high confidentiality may never be output to the network in plain, and values of high integrity are guaranteed not to originate from the attacker. Pair types $T * T'$ describe the type of their components and the type $T \vee T'$ is given to messages that can have type T or type T'.

The type $\tau_n^{l,a}$ describes nonces and constants of security level l: the label a ranges over $\{\infty, 1\}$, denoting whether the nonce is bound within a replication or not (constants are always typed with $a = 1$). We assume a different identifier n for each constant and restriction in the process. The type $\tau_n^{l,1}$ is populated by a single name, (i.e., n describes a constant or a non-replicated nonce) and $\tau_n^{l,\infty}$ is a special type, that is instantiated to $\tau_{n_j}^{l,1}$ in the jth replication of the process. Type $[\![\tau_n^{l,a}\ ;\ \tau_m^{l',a}]\!]$ is a refinement type that restricts the set of possible values of a message to values of type $\tau_n^{l,a}$ on the left and type $\tau_m^{l',a}$ on the right. For a refinement type $[\![\tau_n^{l,a}\ ;\ \tau_n^{l,a}]\!]$ with equal types on both sides we write $\tau_n^{l,a}$.

Keys can have three different types ranged over by KT, ordered by a subtyping relation (SEqKey, SSesKey): $\mathrm{seskey}^{l,a}(T) <: \mathrm{eqkey}^l(T) <: \mathrm{key}^l(T)$. For all three types, l denotes the security label (SKey) of the key and T is the type of the payload that can be encrypted or signed with these keys. This allows us to transfer typing information from one process to another one: e.g. when encrypting, we check that the payload type is respected, so that we can be sure to get a value of the payload type upon decryption. The three different types encode different relations between the left and the right component of a bikey (k, k'). While type $\mathrm{key}^l(T)$ can be given to bikeys with different components $k \neq k'$, type $\mathrm{eqkey}^l(T)$ ensures that the keys are equal on both sides in the specific typed instruction. Type $\mathrm{seskey}^{l,a}(T)$ additionally guarantees that the key is always the same on the left and the right throughout the whole process. We allow for dynamic generation of keys of type $\mathrm{seskey}^{l,a}(T)$ and use a label a to denote whether the key is generated under replication or not – just like for nonce types.

For a key of type T, we use types $\mathrm{pkey}(T)$ and $\mathrm{vkey}(T)$ for the corresponding public key and verification key, and types $(T')_T$ and $\{T'\}_T$ for symmetric and asymmetric encryptions of messages of type T' with this key. Public keys and verification keys can

$$\frac{\Gamma(n) = \tau_n^{l,a} \quad \Gamma(m) = \tau_m^{l,a} \quad l \in \{\mathtt{HH}, \mathtt{HL}\}}{\Gamma \vdash n \sim m : l \to \emptyset}\ (\textsc{TNonce}) \qquad \frac{\Gamma(n) = \tau_n^{\mathtt{LL},a}}{\Gamma \vdash n \sim n : \mathtt{LL} \to \emptyset}\ (\textsc{TNonceL})$$

$$\frac{\Gamma(x) = T}{\Gamma \vdash x \sim x : T \to \emptyset}\ (\textsc{TVar}) \qquad \frac{\Gamma \vdash M \sim N : T' \to c \qquad T' <: T}{\Gamma \vdash M \sim N : T \to c}\ (\textsc{TSub})$$

$$\frac{\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M' \sim N' : T' \to c'}{\Gamma \vdash \langle M, M' \rangle \sim \langle N, N' \rangle : T * T' \to c \cup c'}\ (\textsc{TPair})$$

$$\frac{M, N \text{ well formed}}{\Gamma \vdash M \sim N : \mathtt{HL} \to \emptyset}\ (\textsc{THigh})$$

$$\frac{\Gamma(k, k') = T}{\Gamma \vdash k \sim k' : T \to \emptyset}\ (\textsc{TKey}) \qquad \frac{k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}}{\Gamma \vdash \mathtt{pk}(k) \sim \mathtt{pk}(k) : \mathtt{LL} \to \emptyset}\ (\textsc{TPubKeyL})$$

$$\frac{\Gamma \vdash M \sim N : T \to \emptyset \qquad \exists T', l.T <: \mathrm{key}^l(T')}{\Gamma \vdash \mathtt{pk}(M) \sim \mathtt{pk}(N) : \mathrm{pkey}(T) \to \emptyset}\ (\textsc{TPubKey})$$

$$\frac{\begin{array}{c}\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M' \sim N' : T' \to c' \\ T' = \mathtt{LL} \ \vee \ (\exists T'', T''', l.T' = \mathrm{pkey}(T'') \wedge T'' <: \mathrm{key}^l(T'''))\end{array}}{\Gamma \vdash \mathtt{aenc}(M, M') \sim \mathtt{aenc}(N, N') : \{T\}_{T'} \to c \cup c'}\ (\textsc{TAenc})$$

$$\frac{\Gamma \vdash M \sim N : \{T\}_{\mathrm{pkey}(T')} \to c \qquad T' <: \mathrm{key}^{\mathtt{HH}}(T)}{\Gamma \vdash M \sim N : \mathtt{LL} \to c \cup \{M \sim N\}}\ (\textsc{TAencH})$$

$$\frac{\Gamma \vdash M \sim N : \{\mathtt{LL}\}_T \to c \qquad (T = \mathrm{pkey}(T') \wedge T' <: \mathrm{eqkey}^l(T'')) \text{ or } T = \mathtt{LL}}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}\ (\textsc{TAencL})$$

Fig. 6. Selected rules for messages

be treated as LL if the corresponding keys are equal (SPUBKEY,SVKEY) and subtyping on encryptions is directly induced by subtyping of the payload types (SENC, SAENC).

Constraints When typing messages, we generate constraints of the form $(M \sim N)$, meaning that the attacker may see M and N in the left and right process, respectively, and these two messages are thus required to be indistinguishable.

Due to space reasons we only present a few selected rules that are characteristic of the typing of branching protocols. The omitted rules are similar in spirit to the presented ones or are standard rules for equivalence typing [28].

4.1 Typing messages

The typing judgement for messages is of the form $\Gamma \vdash M \sim N : T \to c$ which reads as follows: under the environment Γ, M and N are of type T and either this is a high confidentiality type (i.e., M and N are not disclosed to the attacker) or M and N are indistinguishable for the attacker assuming the set of constraints c is consistent.

Confidential nonces can be given their label from the typing environment in rule TNONCE. Since their label prevents them from being released in clear, the attacker cannot

observe them and we do not need to add constraints for them. They can however be output in encrypted form and will then appear in the constraints of the encryption. Public nonces (labeled as LL) can be typed if they are equal on both sides (rule TNONCEL). These are standard rules, as well as the rules TVAR, TSUB, TPAIR and THIGH [28].

A non-standard rule that is crucial for the typing of branching protocols is rule TKEY. As the typing environment contains types for bikeys (k, k') this rule allows us to type two potentially different keys with their type from the environment. With the standard rule TPUBKEYL we can only type a public key of the same keys on both sides, while rule TPUBKEY allows us to type different public keys $\mathtt{pk}(M), \mathtt{pk}(N)$, provided we can show that there exists a valid key type for the terms M and N. This highlights another important technical contribution of this work, as compared to existing type systems for equivalence: we do not only support a fixed set of keys, but also allow for the usage of keys in variables, that have been received from the network.

To show that a message is of type $\{T\}_{T'}$ – a message of type T encrypted asymmetrically with a key of type T', we have to show that the corresponding terms have exactly these types in rule TAENC. The generated constraints are simply propagated. In addition we need to show that T' is a valid type for a public key, or LL, which models untrusted keys received from the network. Note, that this rule allows us to encrypt messages with different keys in the two processes. For encryptions with honest keys (label HH) we can use rule TAENC to give type LL to the messages, if we can show that the payload type is respected. In this case we add the entire encryptions to the constraints, since the attacker can check different encryptions for equality, even if he cannot obtain the plaintext. Rule TAENCL allows us to give type LL to encryptions even if we do not respect the payload type, or if the key is corrupted. However, we then have to type the plaintexts with type LL since we cannot guarantee their confidentiality. Additionally, we have to ensure that the same key is used in both processes, because the attacker might possess the corresponding private keys and test which decryption succeeds. Since we already add constraints for giving type LL to the plaintext, we do not need to add any additional constraints.

4.2 Typing processes

From now on, we assume that processes assign a type to freshly generated nonces and keys. That is, $\mathtt{new}\ n.P$ is now of the form $\mathtt{new}\ n : T.\ P$. This requires a (very light) type annotation from the user. The typing judgement for processes is of the form $\Gamma \vdash P \sim Q \to C$ and can be interpreted as follows: If two processes P and Q can be typed in Γ and if the generated constraint set C is consistent, then P and Q are trace equivalent. We present selected rules in Fig. 7.

Rule POUT states that we can output messages to the network if we can type them with type LL, i.e., they are indistinguishable to the attacker, provided that the generated set c of constraints is consistent. The constraints of c are then added to all constraints in the constraint set C. We define $C \cup_\forall c' := \{(c \cup c', \Gamma) \mid (c, \Gamma) \in C\}$. This rule, as well as the rules PZERO, PIN, PNEW, PPAR, and PLET, are standard rules [28].

Rule PNEWKEY allows us to generate new session keys at runtime, which models security protocols more faithfully. It also allows us to generate infinitely many keys, by introducing new keys under replication.

$$\frac{\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c}{\Gamma \vdash \mathtt{out}(M).P \sim \mathtt{out}(N).Q \to C \cup_\forall c} \text{ (POUT)}$$

$$\frac{\Gamma \vdash \diamond \quad \Gamma \text{ does not contain union types}}{\Gamma \vdash \mathtt{0} \sim \mathtt{0} \to (\emptyset, \Gamma)} \text{ (PZERO)} \qquad \frac{\Gamma, x : \mathtt{LL} \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{in}(x).P \sim \mathtt{in}(x).Q \to C} \text{ (PIN)}$$

$$\frac{\Gamma, n : \tau_n^{l,a} \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{new}\ n : \tau_n^{l,a}.P \sim \mathtt{new}\ n : \tau_n^{l,a}.Q \to C} \text{ (PNEW)}$$

$$\frac{\Gamma, (k,k) : \mathrm{seskey}^{l,a}(T) \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{new}\ k : \mathrm{seskey}^{l,a}(T).P \sim \mathtt{new}\ k : \mathrm{seskey}^{l,a}(T).Q \to C} \text{ (PNEWKEY)}$$

$$\frac{\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C'}{\Gamma \vdash P \mid P' \sim Q \mid Q' \to C \cup_\times C'} \text{ (PPAR)}$$

$$\frac{\Gamma \vdash_d t \sim t' : T \qquad \Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C'}{\Gamma \vdash \mathtt{let}\ x = t\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = t'\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to C \cup C'} \text{ (PLET)}$$

(PLETADECSAME)

$$\frac{\begin{array}{c}\Gamma(y) = \mathtt{LL} \qquad \Gamma(k,k) <: \mathrm{key}^{\mathtt{HH}}(T) \\ \Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma, x : \mathtt{LL} \vdash P \sim Q \to C' \qquad \Gamma \vdash P' \sim Q' \to C'' \\ (\forall T'. \forall k' \neq k.\ \Gamma(k,k') <: \mathrm{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P \sim Q' \to C_{k'}) \\ (\forall T'. \forall k' \neq k.\ \Gamma(k',k) <: \mathrm{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P' \sim Q \to C'_{k'})\end{array}}{\begin{array}{c}\Gamma \vdash \mathtt{let}\ x = \mathtt{adec}(y,k)\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = \mathtt{adec}(y,k)\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \\ \to C \cup C' \cup C'' \cup (\bigcup_{k'} C_{k'}) \cup (\bigcup_{k'} C'_{k'})\end{array}}$$

$$\frac{\begin{array}{ccc}\Gamma \vdash P \sim Q \to C_1 & & \\ \Gamma \vdash P \sim Q' \to C_2 & \Gamma \vdash P' \sim Q \to C_3 & \Gamma \vdash P' \sim Q' \to C_4\end{array}}{\begin{array}{c}\Gamma \vdash \mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = N'\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \\ \to C_1 \cup C_2 \cup C_3 \cup C_4\end{array}} \text{ (PIFALL)}$$

Fig. 7. Selected rules for processes

Rule PLETADECSAME treats asymmetric decryptions where we use the same fixed honest key (label HH) for decryptions in both processes. Standard type systems for equivalence have a simplifying (and restrictive) invariant that guarantees that encryptions are always performed using the same keys in both processes and hence guarantee that both processes always take the same branch in decryption (compare rule PLET). In our system however, we allow encryptions with potentially different keys, which requires cross-case validation in order to retain soundness. Still, the number of possible combinations of encryption keys is limited by the assignments in the typing environment Γ. To cover all the possibilities, we type the following combinations of continuation processes:

- Both `then` branches: In this case we know that key k was used for encryption on both sides. Because of $\Gamma(k,k) = \mathrm{key}^{\mathtt{HH}}(T)$, we know that in this case the payload type is T and we type the continuation with $\Gamma, x : T$.
  Because the message may also originate from the attacker (who also has access to the public key), we have to type the two `then` branches also with $\Gamma, x : \mathtt{LL}$.

$$\frac{\Gamma(k,k) <: \text{key}^{\texttt{LL}}(T) \qquad \Gamma(x) = \texttt{LL}}{\Gamma \vdash_d \texttt{adec}(x,k) \sim \texttt{adec}(x,k) : \texttt{LL}} \text{ (DADECL)}$$

$$\frac{\Gamma(y) = \text{seskey}^{\texttt{HH},a}(T) \qquad \Gamma(x) = \texttt{LL}}{\Gamma \vdash_d \texttt{adec}(x,y) \sim \texttt{adec}(x,y) : T \vee \texttt{LL}} \text{ (DADECH')}$$

$$\frac{(\Gamma(y) = \text{seskey}^{\texttt{LL},a}(T) \vee \Gamma(y) = \texttt{LL}) \qquad \Gamma(x) = \texttt{LL}}{\Gamma \vdash_d \texttt{adec}(x,y) \sim \texttt{adec}(x,y) : \texttt{LL}} \text{ (DADECL')}$$

$$\frac{\Gamma(k,k) = \text{seskey}^{l,a}(T') \qquad \Gamma(x) = \{T\}_{\text{pkey}(\text{seskey}^{l,a}(T'))}}{\Gamma \vdash_d \texttt{adec}(x,k) \sim \texttt{adec}(x,k) : T} \text{ (DADECT)}$$

$$\frac{\Gamma(y) = \text{seskey}^{l,a}(T') \qquad \Gamma(x) = \{T\}_{\text{pkey}(\text{seskey}^{l,a}(T'))}}{\Gamma \vdash_d \texttt{adec}(x,y) \sim \texttt{adec}(x,y) : T} \text{ (DADECT')}$$

Fig. 8. Selected destructor rules

- Both `else` branches: If decryption fails on both sides, we type the two `else` branches without introducing any new variables.
- Left `then`, right `else`: The encryption may have been created with key k on the left side and another key k' on the right side. Hence, for each $k' \neq k$, such that $\Gamma(k,k')$ maps to a key type with label `HH` and payload type T', we have to typecheck the left `then` branch and the right `else` branch with $\Gamma, x : T'$.
- Left `else`, right `then`: This case is analogous to the previous one.

The generated set of constraints is simply the union of all generated constraints for the subprocesses. Rule PIFALL lets us typecheck any conditional by simply checking the four possible branch combinations. In contrast to the other rules for conditionals that we present in Appendix A, this rule does not require any other preconditions or checks on the terms M, M', N, N'.

Destructor Rules The rule PLET requires that a destructor application succeeds or fails equally in the two processes. To ensure this property, it relies on additional rules for destructors. We present selected rules in Fig. 8. Rule DADECL is a standard rule that states that a decryption of a variable of type `LL` with an untrusted key (label `LL`) yields a result of type `LL`. Decryption with a trusted (label `HH`) session key gives us a value of the key's payload type or type `LL` in case the encryption was created by the attacker using the public key. Here it is important that the key is of type $\text{seskey}^{\texttt{HH},a}(T)$, since this guarantees that the key is never used in combination with a different key and hence decryption will always equally succeed or fail in both processes. Rule DADECL' is similar to rule DADECL except it uses a variable for decryption instead of a fixed key. Rule DADECT treats the case in which we know that the variable x is an asymmetric encryption of a specific type. If the type of the key used for decryption matches the key type used for encryption, we know the exact type of the result of a successful decryption. DADECT' is similar to DADECT, with a variable as key. In Appendix A we present similar rules for symmetric decryption and verification of signatures.

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, N_b \text{ well formed}}{\Gamma \vdash \langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle \sim N_b : \mathtt{HL} \to \emptyset}\ \textsc{THigh}
\quad
\dfrac{\Gamma(k_a, k) = \mathrm{key}^{\mathtt{HH}}(\mathtt{HL})}{\Gamma \vdash k_a \sim k : \mathrm{key}^{\mathtt{HH}}(\mathtt{HL}) \to \emptyset}\ \textsc{TKey}
}{\Gamma \vdash \mathtt{pk}(k_a) \sim \mathtt{pk}(k) : \mathrm{pkey}(\mathrm{key}^{\mathtt{HH}}(\mathtt{HL})) \to \emptyset}\ \textsc{TPubKey}
}{\Gamma \vdash \mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k)) : \{\mathtt{HL}\}_{\mathrm{pkey}(\mathrm{key}^{\mathtt{HH}}(\mathtt{HL}))} \to \emptyset}\ \textsc{TAenc}
}{\Gamma \vdash \mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k)) : \mathtt{LL} \to C}\ \textsc{TAenCH}
$$

where $C = \{\mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k))\}$.

Fig. 9. Type derivation for the response to A and the decoy message

4.3 Typing the private authentication protocol

We now show how our type system can be applied to type the Private Authentication protocol presented in section 2.3, by showing the most interesting parts of the derivation. We type the protocol using the initial environment Γ presented in Fig. 1.

We focus on the responder process P_b and start with the asymmetric decryption. As we use the same key k_b in both processes, we apply rule PLetADecSame. We have $\Gamma(x) = \mathtt{LL}$ by rule PIn and $\Gamma(k_b, k_b) = \mathrm{key}^{\mathtt{HH}}(\mathtt{HH}, \mathtt{LL})$. We do not have any other entry using key k_b in Γ. We hence typecheck the two `then` branches once with $\Gamma, y : (\mathtt{HH} * \mathtt{LL})$ and once with $\Gamma, y : \mathtt{LL}$, as well as the two `else` branches (which are just `0` in this case).

Typing the let expressions is straightforward using rule PLet. In the conditional we check $y_2 = \mathtt{pk}(k_a)$ in the left process and $y_2 = \mathtt{pk}(k_c)$ in the right process. Since we cannot guarantee which branches are taken or even if the same branch is taken in the two processes, we use rule PIfAll to typecheck all four possible combinations of branches. We now focus on the case where A is successfully authenticated in the left process and is rejected in the right process. We then have to typecheck B's positive answer together with the decoy message: $\Gamma \vdash \mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_c, \mathtt{pk}(k)) : \mathtt{LL}$.

Fig. 9 presents the type derivation for this example. We apply rule TAenc to give type LL to the two terms, adding the two encryptions to the constraint set. Using rule TAencH we can show that the encryptions are well-typed with type $\{\mathtt{HL}\}_{\mathrm{pkey}(\mathrm{key}^{\mathtt{HH}}(\mathtt{HL}))}$. The type of the payload is trivially shown with rule THigh. To type the public key, we use rule TPubKey followed by rule TKey, which looks up the type for the bikey (k_a, k) in the typing environment Γ.

5 Consistency

Our type system collects constraints that intuitively correspond to (symbolic) messages that the attacker may see (or deduce). Therefore, two processes are in trace equivalence only if the collected constraints are in static equivalence for any plausible instantiation.

However, checking static equivalence of symbolic frames for any instantiation corresponding to a real execution may be as hard as checking trace equivalence [24]. Conversely, checking static equivalence for *any* instantiation may be too strong and may prevent proving equivalence of processes. Instead, we use again the typing information

gathered by our type system and we consider only instantiations that comply with the type. Actually, we even restrict our attention to instantiations where variables of type LL are only replaced by deducible terms. This last part is a key ingredient for considering processes with dynamic keys. Hence, we define a constraint to be *consistent* if the corresponding two frames are in static equivalence for any instantiation that can be typed and produces constraints that are included in the original constraint.

Formally, we first introduce the following ingredients:

- $\phi_\ell(c)$ and $\phi_r(c)$ denote the frames that are composed of the left and the right terms of the constraints respectively (in the same order).
- $\phi_{\mathtt{LL}}^{\Gamma}$ denotes the frame that is composed of all low confidentiality nonces and keys in Γ, as well as all public encryption keys and verification keys in Γ. This intuitively corresponds to the initial knowledge of the attacker.
- Two ground substitutions σ, σ' are well-typed in Γ with constraint c_σ if they preserve the types for variables in Γ, *i.e.*, for all x, $\Gamma \vdash \sigma(x) \sim \sigma'(x) : \Gamma(x) \to c_x$, and $c_\sigma = \bigcup_{x \in \mathrm{dom}(\Gamma)} c_x$.

The instantiation of a constraint is defined as expected. If c is a set of constraints, and σ, σ' are two substitutions, let $[\![c]\!]_{\sigma,\sigma'}$ be the instantiation of c by σ on the left and σ' on the right, that is, $[\![c]\!]_{\sigma,\sigma'} = \{M\sigma \sim N\sigma' \mid M \sim N \in c\}$.

Definition 3 (Consistency). *A set of constraints c is* consistent *in an environment Γ if for all substitutions σ,σ' well-typed in Γ with a constraint c_σ such that $c_\sigma \subseteq [\![c]\!]_{\sigma,\sigma'}$, the frames $\phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell(c)\sigma$ and $\phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r(c)\sigma'$ are statically equivalent. We say that (c, Γ) is consistent if c is consistent in Γ and that a constraint set C is consistent in Γ if each element $(c, \Gamma) \in C$ is consistent.*

Compared to [28], we now require $c_\sigma \subseteq [\![c]\!]_{\sigma,\sigma'}$. This means that instead of considering any (well typed) instantiations, we only consider instantiations that use fragments of the constraints. For example, this now imposes that low variables are instantiated by terms deducible from the constraint. This refinement of consistency provides a tighter definition and is needed for non fixed keys, as explained in the next section.

6 Soundness

In this section, we provide our main results. First, soundness of our type system: whenever two processes can be typed with consistent constraints, then they are in trace equivalence. Then we show how to automatically prove consistency. Finally, we explain how to lift these two first results from finite processes to processes with replication. But first, we discuss why we cannot directly apply the results from [28] developed for processes with long term keys.

6.1 Example

Consider the following example, typical for a key-exchange protocol: Alice receives some key and uses it to encrypt, e.g. a nonce. Here, we consider a semi-honest session,

where an honest agent A is receiving a key from a dishonest agent D. Such sessions are typically considered in combination with honest sessions.

$$C \to A : \mathtt{aenc}(\langle k, C\rangle, \mathtt{pk}(A))$$
$$A \to C : \mathtt{aenc}(n, k)$$

The process modelling the role of Alice is as follows.

$$P_A = \mathtt{in}(x).\ \mathtt{let}\ x' = \mathtt{adec}(x, k_A)\ \mathtt{in}\ \mathtt{let}\ y = \pi_1(x')\ \mathtt{in}\ \mathtt{let}\ z = \pi_2(x')\ \mathtt{in}$$
$$\mathtt{if}\ z = C\ \mathtt{then}\ \mathtt{new}\ n.\ \mathtt{out}(\mathtt{enc}(n, y))$$

When type-checking $P_A \sim P_A$ (as part as a more general process with honest sessions), we would collect the constraint $\mathtt{enc}(n, y) \sim \mathtt{enc}(n, y)$ where y comes from the adversary and is therefore a low variable (that is, of type LL). The approach of [28] consisted in opening messages as much as possible. In this example, this would yield the constraint $y \sim y$ which typically renders the constraint inconsistent, as exemplified below.

When typechecking the private authentication protocol, we obtain constraints containing $\mathtt{aenc}(\langle y_1, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k))$ (as seen in Fig. 9), where y_1 has type HL. Assume now that the constraint also contains $y \sim y$ for some variable y of type LL and consider the following instantiations of y and y_1: $\sigma(y_1) = \sigma'(y_1) = a$ for some constant a and $\sigma(y) = \sigma'(y) = \mathtt{aenc}(N_b, \mathtt{pk}(k))$. Note that such an instantiation complies with the type since $\Gamma \vdash \sigma(y) \sim \sigma'(y) : \mathtt{LL} \to c$ for some constraint c. The instantiated constraint would then contain

$$\{\mathtt{aenc}(\langle a, \langle N_b, \mathtt{pk}(k_b)\rangle\rangle, \mathtt{pk}(k_a)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k)),$$
$$\mathtt{aenc}(N_b, \mathtt{pk}(k)) \sim \mathtt{aenc}(N_b, \mathtt{pk}(k))\}$$

and the corresponding frames are not statically equivalent, which makes the constraint inconsistent for the consistency definition of [28] .

Therefore, our first idea consists in proving that we only collect constraints that are saturated w.r.t. deduction: any deducible subterm can already be constructed from the terms of the constraint. Second, we show that for any execution, low variables are instantiated by terms deducible from the constraints. This guarantees that our new notion of consistency is sound. The two results are reflected in the next section.

6.2 Soundness

Our type system, together with consistency, implies trace equivalence.

Theorem 1 (Typing implies trace equivalence). *For all P, Q, and C, for all Γ containing only keys, if $\Gamma \vdash P \sim Q \to C$ and C is consistent, then $P \approx_t Q$.*

Example 3. We can typecheck PA, that is

$$\Gamma \vdash P_a(k_a, \mathtt{pk}(k_b)) \ \mid\ P_b(k_b, \mathtt{pk}(k_a)) \sim P_a(k_a, \mathtt{pk}(k_b)) \ \mid\ P_b(k_b, \mathtt{pk}(k_c)) \to C_{PA}$$

where Γ has been defined in Fig. 1 and assuming that nonce N_a of process P_a has been annotated with type $\tau_{N_a}^{\mathtt{HH},1}$ and nonce N_b of P_b has been annotated with type $\tau_{N_b}^{\mathtt{HH},1}$. The

constraint set C_{PA} can be proved to be consistent using the procedure presented in the next section. Therefore, we can conclude that

$$P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_a)) \approx_t P_a(k_a, \mathtt{pk}(k_b)) \mid P_b(k_b, \mathtt{pk}(k_c))$$

which shows anonymity of the private authentication protocol.

The first key ingredient in the proof of Theorem 1 is the fact that any well-typed low term is deducible from the constraint generated when typing it.

Lemma 1 (Low terms are recipes on their constraints). *For all ground messages M, N, for all Γ, c, if $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ then there exists an attacker recipe R without destructors such that $M = R(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}})$ and $N = R(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$.*

The second key ingredient is a finer invariant on protocol executions: for any typable pair of processes P, Q, any execution of P can be mimicked by an execution of Q such that low variables are instantiated by well-typed terms constructible from the constraint.

Lemma 2. *For all processes P, Q, for all ϕ, σ, for all multisets of processes $\mathcal{P}$, constraint sets C, sequences s of actions, for all Γ containing only keys, if $\Gamma \vdash P \sim Q \to C$, C is consistent, and $(\{P\}, \emptyset, \emptyset) \xrightarrow{s}_* (\mathcal{P}, \phi, \sigma)$, then there exist a sequence s' of actions, a multiset $\mathcal{Q}$, a frame ϕ', a substitution σ', an environment Γ', a constraint c such that:*
– $(\{Q\}, \emptyset, \emptyset) \xrightarrow{s'}_* (\mathcal{Q}, \phi', \sigma')$, with $s =_\tau s'$
– $\Gamma' \vdash \phi\sigma \sim \phi'\sigma' : \mathtt{LL} \to c$, and for all $x \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma')$, there exists c_x such that $\Gamma' \vdash \sigma(x) \sim \sigma(x) : \Gamma'(x) \to c_x$ and $c_x \subseteq c$.

Note that this finer invariant guarantees that we can restrict our attention to the instantiations considered for defining consistency.

As a by-product, we obtain a finer type system for equivalence, even for processes with long term keys (as in [28]). For example, we can now prove equivalence of processes where some agent signs a low message that comes from the adversary. In such a case, we collect $\mathtt{sign}(x, k) \sim \mathtt{sign}(x, k)$ in the constraint, where x has type $\mathtt{LL}$, which we can now prove to be consistent (depending on how x is used in the rest of the constraint).

6.3 Procedure for consistency

We devise a procedure $\mathtt{check\_const}(C)$ for checking consistency of a constraint C, depicted in Figure 10. Compared to [28], the procedure is actually simplified. Thanks to Lemmas 1 and 2, there is no need to open constraints anymore. The rest is very similar and works as follows:

- First, variables of refined type $[\![\tau_m^{l,1} ; \tau_n^{l',1}]\!]$ are replaced by m on the left-hand-side of the constraint and n on the right-hand-side.
- Second, we check that terms have the same shape (encryption, signature, hash) on the left and on the right and that asymmetric encryption and hashes cannot be reconstructed by the adversary (that is, they contain some fresh nonce).
- The most important step consists in checking that the terms on the left satisfy the same equalities than the ones on the right. Whenever two left terms M and N are unifiable, their corresponding right terms M' and N' should be equal after applying a similar instantiation.

$\texttt{step1}_\Gamma(c) := (\llbracket c \rrbracket_{\sigma_F, \sigma'_F}, \Gamma')$, with

$$F := \{x \in \mathrm{dom}(\Gamma) \mid \exists m, n, l, l'.\ \Gamma(x) = \llbracket \tau_m^{l,1}\ ;\ \tau_n^{l',1} \rrbracket\}$$

and σ_F, σ'_F defined by

$$\begin{cases} \bullet\ \mathrm{dom}(\sigma_F) = \mathrm{dom}(\sigma'_F) = F \\ \bullet\ \forall x \in F.\ \forall m, n, l, l'. \Gamma(x) = \llbracket \tau_m^{l,1}\ ;\ \tau_n^{l',1} \rrbracket \Rightarrow \sigma_F(x) = m\ \wedge\ \sigma'_F(x) = n \end{cases}$$

and Γ' is $\Gamma|_{\mathrm{dom}(\Gamma)\setminus F}$ extended with $\Gamma'(n) = \tau_n^{l,1}$ for all nonce n such that $\tau_n^{l,1}$ occurs in Γ.

$\texttt{step2}_\Gamma(c) :=$ check that for all $M \sim N \in c$, M and N are both

- $\texttt{enc}(M', M'')$, $\texttt{enc}(N', N'')$ where M'', N'' are either
  - keys k, k' where $\exists T. \Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$;
  - or a variable x such that $\exists T. \Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$;
- or encryptions $\texttt{aenc}(M', M'')$, $\texttt{aenc}(N', N'')$ where
  - M' and N' contain directly under pairs a nonce n such that $\Gamma(n) = \tau_n^{\mathtt{HH},a}$ or a secret key k such that $\exists T, k'. \Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$ or $\Gamma(k', k) <: \mathrm{key}^{\mathtt{HH}}(T)$, or a variable x such that $\exists m, n, a. \Gamma(x) = \llbracket \tau_m^{\mathtt{HH},a}\ ;\ \tau_n^{\mathtt{HH},a} \rrbracket$, or a variable x such that $\exists T. \Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$;
  - M'' and N'' are either
    - public keys $\texttt{pk}(k)$, $\texttt{pk}(k')$ where $\exists T. \Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$;
    - or public keys $\texttt{pk}(x)$, $\texttt{pk}(x)$ where $\exists T. \Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$;
    - or a variable x such that $\exists T, T'. \Gamma(x) = \mathrm{pkey}(T)$ and $T <: \mathrm{key}^{\mathtt{HH}}(T')$;
- or hashes $\texttt{h}(M')$, $\texttt{h}(N')$, where M', N' similarly contain a secret value under pairs;
- or signatures $\texttt{sign}(M', M'')$, $\texttt{sign}(N', M'')$ where M'', N'' are either
  - keys k, k' where $\exists T. \Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$;
  - or a variable x such that $\exists T. \Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$;

$\texttt{step3}_\Gamma(c) :=$ If for all $M \sim M'$ and $N \sim N' \in c$ such that M, N are unifiable with a most general unifier μ, and such that

$$\forall x \in \mathrm{dom}(\mu). \exists l, l', m, p.\ (\Gamma(x) = \llbracket \tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty} \rrbracket) \Rightarrow (x\mu \in \mathcal{X}\ \vee\ \exists i.\ x\mu = m_i)$$

we have

$$M'\alpha\theta = N'\alpha\theta$$

where

$$\forall x \in \mathrm{dom}(\mu). \forall l, l', m, p, i. (\Gamma(x) = \llbracket \tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty} \rrbracket\ \wedge\ \mu(x) = m_i) \Rightarrow \theta(x) = p_i$$

and α is the restriction of μ to $\{x \in \mathrm{dom}(\mu) \mid \Gamma(x) = \mathtt{LL}\ \wedge\ \mu(x) \in \mathcal{N}\}$;
and if the symmetric condition for the case where M', N' are unifiable holds as well, then return `true`.

$\texttt{check\_const}(C) :=$ for all $(c, \Gamma) \in C$, let $(c_1, \Gamma_1) := \texttt{step1}_\Gamma(c)$ and check that $\texttt{step2}_{\Gamma_1}(c_1) = \texttt{true}$ and $\texttt{step3}_{\Gamma_1}(c_1) = \texttt{true}$.

Fig. 10. Procedure for checking consistency.

For constraint sets without infinite nonce types, `check_const` entails consistency.

Theorem 2. *Let C be a set of constraints such that*

$$\forall (c, \Gamma) \in C.\ \forall l, l', m, p.\ \Gamma(x) \neq [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!].$$

If $\texttt{check\_const}(C) = \texttt{true}$*, then C is consistent.*

Example 4. Continuing Example 3, typechecking the PA protocol yields the set C_{PA} of constraint sets. C_{PA} contains in particular the set

$$\begin{aligned} \{\ &\texttt{aenc}(\langle N_a, \texttt{pk}(k_a)\rangle, \texttt{pk}(k_b)) \sim \texttt{aenc}(\langle N_a, \texttt{pk}(k_a)\rangle, \texttt{pk}(k_b)), \\ &\texttt{aenc}(\langle y_1, \langle N_b, \texttt{pk}(k_b)\rangle\rangle, \texttt{pk}(k_a)) \sim \texttt{aenc}(N_b, \texttt{pk}(k))\ \} \end{aligned}$$

where variable y_1 has type `HL` (we also have the same constraint but where y_1 has type `LL`). The other constraint sets of C_{PA} are similar and correspond to the various cases (`else` branch of P_a with `then` branch of P_b, etc.). The procedure `check_const` returns true since no two terms can be unified, which proves consistency. Similarly, the other constraints generated for PA can be proved to be consistent applying `check_const`.

6.4 From finite to replicated processes

The previous results apply to processes without replication only. In the spirit of [28], we lift our results to replicated processes. We proceed in two steps.

1. Whenever $\Gamma \vdash P \sim Q \to C$, we show that:
   $[\,\Gamma\,]_1 \cup \cdots \cup [\,\Gamma\,]_n \vdash [\,P\,]_1 | \ldots | [\,P\,]_n \sim [\,Q\,]_1 | \ldots | [\,Q\,]_n \to [\,C\,]_1 \cup_\times \cdots \cup_\times [\,C\,]_n$,
   where $[\,\Gamma\,]_i$ is intuitively a copy of Γ, where variables x have been replaced by x_i, and nonces or keys n of infinite type $\tau_n^{l,\infty}$ (or $\text{seskey}^{l,\infty}(T)$) have been replaced by n_i. The copies $[\,P\,]_i$, $[\,Q\,]_i$, and $[\,C\,]_i$ are defined similarly.
2. We cannot directly check consistency of infinitely many constraints of the form $[\,C\,]_1 \cup_\times \cdots \cup_\times [\,C\,]_n$. Instead, we show that it is sufficient to check consistency of two copies $[\,C\,]_1 \cup_\times [\,C\,]_2$ only. The reason why we need two copies (and not just one) is to detect when messages from different sessions may become equal.

Formally, we can prove trace equivalence of replicated processes.

Theorem 3. *Consider P, Q, P' ,Q', C, C', such that P, Q and P', Q' do not share any variable. Consider Γ, containing only keys and nonces with finite types.*

Assume that P and Q only bind nonces and keys with infinite nonce types, i.e. *using* `new` $m : \tau_m^{l,\infty}$ *and* `new` $k : \text{seskey}^{l,\infty}(T)$ *for some label l and type T; while P' and Q' only bind nonces and keys with finite types,* i.e. *using* `new` $m : \tau_m^{l,1}$ *and* `new` $k : \text{seskey}^{l,1}(T)$.

Let us abbreviate by `new` $\overline{n}$ *the sequence of declarations of each nonce $m \in \text{dom}(\Gamma)$ and session key k such that $\Gamma(k,k) = \text{seskey}^{l,1}(T)$ for some l, T. If*

- $\Gamma \vdash P \sim Q \to C$,
- $\Gamma \vdash P' \sim Q' \to C'$,

– $\texttt{check\_const}([\,C\,]_1 \cup_\times [\,C\,]_2 \cup_\times [\,C'\,]_1) = \texttt{true}$,

then $\qquad \texttt{new}\ \overline{n}.\ ((!P) \mid P') \approx_t \texttt{new}\ \overline{n}.\ ((!Q) \mid Q')$.

The proof, together with fully precise definitions, can be found in Appendices A and B.Interestingly, Theorem 3 allows to consider a mix of finite and replicated processes.

7 Experimental results

We implemented our typechecker as well as our procedure for consistency in a prototype tool TypeEq. We adapted the original prototype of [28] to implement additional cases corresponding to the new typing rules. This also required to design new heuristics w.r.t. the order in which typing rules should be applied. Of course, we also had to support for the new bikey types, and for arbitrary terms as keys. This represented a change of about 40% of the code of the software. We ran our experiments on a single Intel Xeon E5-2687Wv3 3.10GHz core, with 378GB of RAM (shared with the 19 other cores). Actually, our own prototype does not require a large amount of RAM. However, some of the other tools we consider use more than 64GB of RAM on some examples (at which point we stopped the experiment). More precise figures about our tool are provided in the table of Figure 11. The corresponding files can be found at [27].

We tested TypeEq on two symmetric key protocols that include a handshake on the key (Yahalom-Lowe and Needham-Schroeder symmetric key protocols). In both cases, we prove key usability of the exchanged key. Intuitively, we show that an attacker cannot distinguish between two encryptions of public constants: $P.\texttt{out}(\texttt{enc}(a,k)) \approx_t P.\texttt{out}(\texttt{enc}(b,k))$. We also consider one standard asymmetric key protocol (Needham-Schroeder-Lowe protocol), showing strong secrecy of the exchanged nonce.

Helios [4] is a well known voting protocol. We show ballot privacy, in the presence of a dishonest board, assuming that voters do not revote (otherwise the protocol is subject to a copy attack [38], a variant of [29]). We consider a more precise model than the previous Helios models which assume that voters initially know the election public key. Here, we model the fact that voters actually receive the (signed) freshly generated election public key from the network. The BAC protocol is one of the protocols embedded in the biometric passport [1]. We show anonymity of the passport holder $P(A) \approx_t P(B)$. Actually, the only data that distinguish $P(A)$ from $P(B)$ are the private keys. Therefore we consider an additional step where the passport sends the identity of the agent to the reader, encrypted with the exchanged key. Finally, we consider the private authentication protocol, as described in this paper.

7.1 Bounded number of sessions

We first compare TypeEq with the tools for a bounded number of sessions. Namely, we consider Akiss [22], APTE [23] as well as its optimised variant with partial order reduction APTE-POR [10], SPEC [31], and SatEquiv [26]. We step by step increase the number of sessions until we reach a "complete" scenario where each role is instantiated by A talking to B, A talking to C, B talking to A, and B talking to C, where A, B

Protocols (# sessions)		Akiss	APTE	APTE-POR	Spec	Sat-Eq	TypeEq	
							Time	Memory
Needham - Schroeder (symmetric)	3	4.2s	0.39s	0.086s	59.3s	0.14s	0.006s	4.0 MB
	6	TO	TO	9m22s	TO	0.53s	0.009s	4.7 MB
	10			SO		3.7s	0.012s	5.0 MB
	14					18s	0.015s	6.9 MB
Yahalom - Lowe	3	1.0s	2.9s	0.095s	10s	0.063s	0.006s	3.8 MB
	6	MO	TO	11m20s	MO	0.26s	0.017s	4.9 MB
	10			SO		3.0s	0.015s	4.9 MB
	14					18s	0.019s	5.0 MB
Needham-Schroeder-Lowe	2	0.10s	3.8s	0.06s	28s	x	0.004s	3.1 MB
	4	1m8s	BUG	BUG	TO		0.004s	3.4 MB
	8	TO					0.007s	4.7 MB
Private Authentication	2	0.19s	1.2s	0.034s	x	x	0.004s	3.2 MB
	4	99m	TO	24.6s			0.013s	4.9 MB
	8	MO		TO			1s	37 MB
Helios	3	MO	BUG	BUG	x	x	0.005s	3.5 MB
BAC	2	4.0s	0.20s	0.032s	x	x	0.004s	2.9 MB
	3	SO	185m	2.6s			0.004s	3.1 MB
	5		TO	107m			0.005s	3.4 MB
	7			TO			0.005s	3.8 MB

TO: Time Out (>12h) MO: Memory Overflow (>64GB) SO: Stack Overflow

Fig. 11. Experimental results for the bounded case

are honest while C is dishonest. This yields 14 sessions for symmetric-key protocols with two agents and one server, and 8 sessions for a protocol with two agents. In some cases, we further increase the number of sessions (replicating identical scenarios) to better compare tools performance. The results of our experiments are reported in Fig. 11. Note that SatEquiv fails to cover several cases because it does not handle asymmetric encryption nor else branches.

7.2 Unbounded number of sessions

We then compare TypeEq with Proverif. As shown in Fig. 12, the performances are similar except that ProVerif cannot prove Helios. The reason lies in the fact that Helios is actually subject to a copy attack if voters revote and ProVerif cannot properly handle processes that are executed only once. Similarly, Tamarin cannot properly handle the else branch of Helios (which models that the ballot box rejects duplicated ballots). Tamarin fails to prove that the underlying check either succeeds or fails on both sides.

8 Conclusion and discussion

We devise a new type system to reason about keys in the context of equivalence properties. Our new type system significantly enhances the preliminary work of [28], covering a

Protocols	ProVerif	TypeEq
Helios	x	0.005s
Needham-Schroeder (sym)	0.23s	0.016s
Needham-Schroeder-Lowe	0.08s	0.008s
Yahalom-Lowe	0.48s	0.020s
Private Authentication	0.034s	0.008s
BAC	0.038s	0.005s

Fig. 12. Experimental results for an unbounded number of sessions

larger class of protocols that includes key-exchange protocols, protocols with setup phases, as well as protocols that branch differently depending on the decryption key.

Our type system requires a light type annotation that can be directly inferred from the structure of the messages. As future work, we plan to develop an automatic type inference system. In our case study, the only intricate case is the Helios protocol where the user has to write a refined type that corresponds to an over-approximation of any encrypted message. We plan to explore whether such types could be inferred automatically.

We also plan to study how to add phases to our framework, in order to cover more properties (such as unlinkability). This would require to generalize our type system to account for the fact that the type of a key may depend on the phase in which it is used.

Another limitation of our type system is that it does not address processes with too dissimilar structure. While our type system goes beyond diff-equivalence, e.g. allowing else branches to be matched with then branches, we cannot prove equivalence of processes where traces of P are dynamically mapped to traces of Q, depending on the attacker's behaviour. Such cases occur for example when proving unlinkability of the biometric passport. We plan to explore how to enrich our type system with additional rules that could cover such cases, taking advantage of the modularity of the type system.

Conversely, the fact that our type system discards processes that are in equivalence shows that our type system proves something stronger than trace equivalence. Indeed, processes P and Q have to follow some form of uniformity. We could exploit this to prove stronger properties like oblivious execution, probably further restricting our typing rules, in order to prove e.g. the absence of side-channels of a certain form.

Acknowledgments

This work has been partially supported by the European Research Council (ERC) under the European Union's Horizon 2020 research (grant agreements No 645865-SPOOC and No 771527-BROWSEC).

References

1. Machine readable travel document. Tech. Rep. 9303, International Civil Aviation Organization (2008)

2. Abadi, M., Fournet, C.: Mobile Values, New Names, and Secure Communication. In: 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'01). pp. 104–115. ACM (2001)
3. Abadi, M., Fournet, C.: Private authentication. Theoretical Computer Science 322(3), 427 – 476 (2004)
4. Adida, B.: Helios: web-based open-audit voting. In: 17th conference on Security symposium. pp. 335–348. SS'08 (2008)
5. Arapinis, M., Chothia, T., Ritter, E., Ryan, M.: Analysing unlinkability and anonymity using the applied pi calculus. In: 2nd IEEE Computer Security Foundations Symposium (CSF'10). IEEE Computer Society Press (2010)
6. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Hankes Drielsma, P., Héam, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA Tool for the automated validation of internet security protocols and applications. In: 17th International Conference on Computer Aided Verification, CAV'2005. LNCS, vol. 3576, pp. 281–285. Springer (2005)
7. Backes, M., Catalin, H., Maffei, M.: Union, Intersection and Refinement Types and Reasoning About Type Disjointness for Secure Protocol Implementations. Journal of Computer Security 22(2), 301–353 (Mar 2014)
8. Backes, M., Hritcu, C., Maffei, M.: Automated Verification of Remote Electronic Voting Protocols in the Applied Pi-Calculus. In: 21st IEEE Computer Security Foundations Symposium. pp. 195–209. CSF '08, IEEE Computer Society (2008)
9. Backes, M., Maffei, M., Unruh, D.: Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In: IEEE Symposium on Security and Privacy. pp. 202–215. SP '08, IEEE Computer Society (2008)
10. Baelde, D., Delaune, S., Hirschi, L.: Partial order reduction for security protocols. In: Proc. 26th International Conference on Concurrency Theory (CONCUR'15). LIPIcs, vol. 42, pp. 497–510. Leibniz-Zentrum für Informatik (2015)
11. Basin, D., Dreier, J., Sasse, R.: Automated Symbolic Proofs of Observational Equivalence. In: 22nd ACM SIGSAC Conference on Computer and Communications Security (ACM CCS 2015). pp. 1144–1155. ACM (Oct 2015)
12. Bengtson, J., Bhargavan, K., Fournet, C., Gordon, A.D., Maffeis, S.: Refinement Types for Secure Implementations. ACM Transactions on Programming Languages and Systems 33(2), 8:1–8:45 (2011)
13. Blanchet, B.: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW-14). pp. 82–96. IEEE Computer Society (Jun 2001)
14. Blanchet, B.: Modeling and verifying security protocols with the applied pi calculus and ProVerif. Foundations and Trends in Privacy and Security 1(1–2), 1–135 (2016)
15. Blanchet, B., Abadi, M., Fournet, C.: Automated verification of selected equivalences for security protocols. Journal of Logic and Algebraic Programming 75(1), 3–51 (Feb–Mar 2008)
16. Bugliesi, M., Calzavara, S., Eigner, F., Maffei, M.: Resource-aware authorization policies for statically typed cryptographic protocols. In: 24th IEEE Computer Security Foundations Symposium. pp. 83–98. CSF '11, IEEE Computer Society (2011)
17. Bugliesi, M., Calzavara, S., Eigner, F., Maffei, M.: Logical foundations of secure resource management in protocol implementations. In: 2nd International Conference on Principles of Security and Trust. pp. 105–125. POST 2013, Springer (2013)
18. Bugliesi, M., Calzavara, S., Eigner, F., Maffei, M.: Affine Refinement Types for Secure Distributed Programming. ACM Transactions on Programming Languages and Systems 37(4), 11:1–11:66 (Aug 2015)

19. Bugliesi, M., Focardi, R., Maffei, M.: Authenticity by tagging and typing. In: 2004 ACM Workshop on Formal Methods in Security Engineering. pp. 1–12. FMSE '04, ACM (2004)
20. Bugliesi, M., Focardi, R., Maffei, M.: Analysis of typed analyses of authentication protocols. In: 18th IEEE Workshop on Computer Security Foundations. pp. 112–125. CSFW '05, IEEE Computer Society (2005)
21. Bugliesi, M., Focardi, R., Maffei, M.: Dynamic types for authentication. Journal of Computer Security 15(6), 563–617 (Dec 2007)
22. Chadha, R., Ciobâcă, S., Kremer, S.: Automated verification of equivalence properties of cryptographic protocols. In: Programming Languages and Systems —Proceedings of the 21th European Symposium on Programming (ESOP'12). LNCS, vol. 7211, pp. 108–127. Springer (Mar 2012)
23. Cheval, V.: Apte: an algorithm for proving trace equivalence. In: Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'14). LNCS, vol. 8413, pp. 587–592 (Apr 2014)
24. Cheval, V., Cortier, V., Delaune, S.: Deciding equivalence-based properties using constraint solving. Theoretical Computer Science 492, 1–39 (2013)
25. Cheval, Vincent and Cortier, Véronique and Plet, Antoine: Lengths may break privacy – or how to check for equivalences with length. In: 25th International Conference on Computer Aided Verification (CAV'13). LNCS, vol. 8043, pp. 708–723. Springer (July 2013)
26. Cortier, V., Delaune, S., Dallon, A.: Sat-equiv: an efficient tool for equivalence properties. In: Proceedings of the 30th IEEE Computer Security Foundations Symposium (CSF'17). IEEE Computer Society Press (August 2017)
27. Cortier, V., Grimm, N., Lallemand, J., Maffei, M.: Typeeq, https://members.loria.fr/JLallemand/files/typing
28. Cortier, V., Grimm, N., Lallemand, J., Maffei, M.: A Type System for Privacy Properties. In: 24th ACM Conference on Computer and Communications Security (CCS'17). pp. 409–423. ACM (2017)
29. Cortier, V., Smyth, B.: Attacking and fixing Helios: An analysis of ballot secrecy. Journal of Computer Security 21(1), 89–148 (2013)
30. Cremers, C.: The Scyther Tool: Verification, falsification, and analysis of security protocols. In: Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, USA, Proc. LNCS, vol. 5123/2008, pp. 414–418. Springer (2008)
31. Dawson, J., Tiu, A.: Automating open bisimulation checking for the spi-calculus. In: IEEE Computer Security Foundations Symposium (CSF 2010) (2010)
32. Delaune, S., Kremer, S., Ryan, M.D.: Verifying privacy-type properties of electronic voting protocols. Journal of Computer Security 17(4), 435–487 (2009)
33. Eigner, F., Maffei, M.: Differential privacy by typing in security protocols. In: 26th IEEE Computer Security Foundations Symposium. pp. 272–286. CSF '13, IEEE Computer Society (2013)
34. Escobar, S., Meadows, C., Meseguer, J.: A rewriting-based inference system for the NRL protocol analyzer and its meta-logical properties. Theoretical Computer Science 367(1–2), 162–202 (2006)
35. Focardi, R., Maffei, M.: Types for Security Protocols. In: Formal Models and Techniques for Analyzing Security Protocols, Cryptology and Information Security Series, vol. 5, chap. 7, pp. 143–181. IOS Press (2011)
36. Gordon, A.D., Jeffrey, A.: Authenticity by typing for security protocols. Journal of Computer Security 11(4), 451–519 (Jul 2003)
37. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: Computer Aided Verification, 25th International Conference, CAV 2013, Princeton, USA, Proc. LNCS, vol. 8044, pp. 696–701. Springer (2013)

38. Roenne, P.: Private communication (2016)
39. Santiago, S., Escobar, S., Meadows, C.A., Meseguer, J.: A Formal Definition of Protocol Indistinguishability and Its Verification Using Maude-NPA. In: STM 2014. pp. 162–177. LNCS, IEEE Computer Society (2014)

A Typing rules and definitions

We give on Figures 13, 14, 15, 16, 17, 18 and 19 a complete version of our types and typing rules, as well as the formal definition of the well-formedness judgement for typing environments.

$$
\begin{array}{lcl}
l & ::= & \mathtt{LL} \mid \mathtt{HL} \mid \mathtt{HH} \\
KT & ::= & \mathrm{key}^{l}(T) \mid \mathrm{eqkey}^{l}(T) \mid \mathrm{seskey}^{l,a}(T) \text{ with } a \in \{1, \infty\} \\
T & ::= & l \mid T * T \mid KT \mid \mathrm{pkey}(KT) \mid \mathrm{vkey}(KT) \\
 & & \mid (T)_T \mid \{T\}_T \\
 & & \mid [\![\tau_n^{l,a} \,;\, \tau_m^{l',a}]\!] \text{ with } a \in \{1, \infty\} \mid T \vee T
\end{array}
$$

Fig. 13. Types for terms

We denote the sets of all keys in Γ by:

$$\mathrm{keys}(\Gamma) = \{k \in \mathcal{K} \mid \exists k' \in \mathcal{K}.\ (k,k') \in \mathrm{dom}(\Gamma) \vee (k',k) \in \mathrm{dom}(\Gamma)\}.$$

We denote the set of all session keys in Γ by:

$$\mathrm{seskeys}(\Gamma) = \{k \in \mathcal{K} \mid \exists l, a, T.\ \Gamma(k,k) = \mathrm{seskey}^{l,a}(T)\}.$$

$$[] \vdash \diamond \ (\textsc{GNil}) \qquad \frac{\Gamma \vdash \diamond \qquad n \in \mathcal{BN} \qquad n \notin \mathrm{dom}(\Gamma)}{\Gamma, n : \tau_n^{l,a} \vdash \diamond} \ (\textsc{GNonce})$$

$$\frac{\Gamma \vdash \diamond \qquad x \notin \mathrm{dom}(\Gamma)}{\Gamma, x : T \vdash \diamond} \ (\textsc{GVar})$$

$$\frac{\begin{array}{c}\Gamma \vdash \diamond \qquad k, k' \in \mathcal{BK} \qquad (k,k') \notin \mathrm{dom}(\Gamma) \qquad k \notin \mathrm{seskeys}(\Gamma) \qquad k' \notin \mathrm{seskeys}(\Gamma) \\ (\forall l, T'.\ \Gamma(k,k) = \mathrm{eqkey}^{l}(T') \ \Rightarrow\ l = \mathtt{HH}) \qquad (\forall l, T'.\ \Gamma(k',k') = \mathrm{eqkey}^{l}(T') \ \Rightarrow\ l = \mathtt{HH})\end{array}}{\Gamma, (k,k') : \mathrm{key}^{\mathtt{HH}}(T) \vdash \diamond} \ (\textsc{GKey})$$

$$\frac{\begin{array}{c}\Gamma \vdash \diamond \qquad k \in \mathcal{BK} \qquad (k,k) \notin \mathrm{dom}(\Gamma) \\ (\forall k', l', T'.\ \Gamma(k,k') <: \mathrm{key}^{l}(T') \ \Rightarrow\ l = l') \qquad (\forall k', l', T'.\ \Gamma(k',k) <: \mathrm{key}^{l}(T') \ \Rightarrow\ l = l')\end{array}}{\Gamma, (k,k) : \mathrm{eqkey}^{l}(T) \vdash \diamond} \ (\textsc{GEqKey})$$

$$\frac{\Gamma \vdash \diamond \qquad k \in \mathcal{BK} \qquad \forall k' \neq k. (k,k') \notin \mathrm{dom}(\Gamma) \ \wedge\ (k',k) \notin \mathrm{dom}(\Gamma)}{\Gamma, (k,k) : \mathrm{seskey}^{l,a}(T) \vdash \diamond} \ (\textsc{GSesKey})$$

Fig. 14. Well-formedness of the typing environment

In this section, we also provide additional definitions (or more precise versions of previous definitions) regarding constraints, and especially their consistency, that the proofs require.

$$\frac{\Gamma(n) = \tau_n^{l,a} \qquad \Gamma(m) = \tau_m^{l,a} \qquad l \in \{\mathtt{HH}, \mathtt{HL}\}}{\Gamma \vdash n \sim m : l \to \emptyset}\ (\textsc{TNonce})$$

$$\frac{\Gamma(n) = \tau_n^{\mathtt{LL},a}}{\Gamma \vdash n \sim n : \mathtt{LL} \to \emptyset}\ (\textsc{TNonceL}) \quad \frac{a \in \mathcal{C} \cup \mathcal{FN} \cup \mathcal{FK}}{\Gamma \vdash a \sim a : \mathtt{LL} \to \emptyset}\ (\textsc{TCstFN})$$

$$\frac{\Gamma \vdash M \sim N : T \to \emptyset \qquad \exists T', l.T <: \mathrm{key}^l(T')}{\Gamma \vdash \mathtt{pk}(M) \sim \mathtt{pk}(N) : \mathrm{pkey}(T) \to \emptyset}\ (\textsc{TPubKey}) \quad \frac{k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}}{\Gamma \vdash \mathtt{pk}(k) \sim \mathtt{pk}(k) : \mathtt{LL} \to \emptyset}\ (\textsc{TPubKeyL})$$

$$\frac{\Gamma \vdash M \sim N : T \to \emptyset \qquad \exists T', l.T <: \mathrm{key}^l(T')}{\Gamma \vdash \mathtt{vk}(M) \sim \mathtt{vk}(N) : \mathrm{vkey}(T) \to \emptyset}\ (\textsc{TVKey}) \quad \frac{k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}}{\Gamma \vdash \mathtt{vk}(k) \sim \mathtt{vk}(k) : \mathtt{LL} \to \emptyset}\ (\textsc{TVKeyL})$$

$$\frac{\Gamma(k, k') = T}{\Gamma \vdash k \sim k' : T \to \emptyset}\ (\textsc{TKey}) \quad \frac{\Gamma(x) = T}{\Gamma \vdash x \sim x : T \to \emptyset}\ (\textsc{TVar})$$

$$\frac{\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M' \sim N' : T' \to c'}{\Gamma \vdash \langle M, M' \rangle \sim \langle N, N' \rangle : T * T' \to c \cup c'}\ (\textsc{TPair})$$

$$\frac{\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M' \sim N' : T' \to c' \qquad T' = \mathtt{LL} \ \vee\ (\exists T'', l.T' <: \mathrm{key}^l(T''))}{\Gamma \vdash \mathtt{enc}(M, M') \sim \mathtt{enc}(N, N') : (T)_{T'} \to c \cup c'}\ (\textsc{TEnc})$$

$$\frac{\Gamma \vdash M \sim N : (T)_{T'} \to c \qquad T' <: \mathrm{key}^{\mathtt{HH}}(T)}{\Gamma \vdash M \sim N : \mathtt{LL} \to c \cup \{M \sim N\}}\ (\textsc{TEncH}) \quad \frac{\Gamma \vdash M \sim N : (\mathtt{LL})_T \to c \qquad T <: \mathrm{key}^{\mathtt{LL}}(T') \text{ or } T = \mathtt{LL}}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}\ (\textsc{TEncL})$$

$$\frac{\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M' \sim N' : T' \to c' \qquad T' = \mathtt{LL} \ \vee\ (\exists T'', T''', l.T' = \mathrm{pkey}(T'') \wedge T'' <: \mathrm{key}^l(T'''))}{\Gamma \vdash \mathtt{aenc}(M, M') \sim \mathtt{aenc}(N, N') : \{T\}_{T'} \to c \cup c'}\ (\textsc{TAenc})$$

$$\frac{\Gamma \vdash M \sim N : \{T\}_{\mathrm{pkey}(T')} \to c \qquad T' <: \mathrm{key}^{\mathtt{HH}}(T)}{\Gamma \vdash M \sim N : \mathtt{LL} \to c \cup \{M \sim N\}}\ (\textsc{TAencH})$$

$$\frac{\Gamma \vdash M \sim N : \{\mathtt{LL}\}_T \to c \qquad (T = \mathrm{pkey}(T') \wedge T' <: \mathrm{eqkey}^l(T'')) \text{ or } T = \mathtt{LL}}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}\ (\textsc{TAencL})$$

$$\frac{\Gamma \vdash M \sim N : T \to c \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c' \qquad \Gamma \vdash M' \sim N' : \mathrm{eqkey}^{\mathtt{HH}}(T) \to \emptyset}{\Gamma \vdash \mathtt{sign}(M, M') \sim \mathtt{sign}(N, N') : \mathtt{LL} \to c \cup c' \cup \{\mathtt{sign}(M, M') \sim \mathtt{sign}(N, N')\}}\ (\textsc{TSignH})$$

$$\frac{\Gamma \vdash M \sim N : \mathtt{LL} \to c \qquad \Gamma \vdash M' \sim N' : \mathtt{LL} \to c'}{\Gamma \vdash \mathtt{sign}(M, M') \sim \mathtt{sign}(N, N') : \mathtt{LL} \to c \cup c'}\ (\textsc{TSignL})$$

$$\frac{\Gamma \vdash M \sim N : \mathtt{HL} \to \emptyset}{\Gamma \vdash \mathtt{h}(M) \sim \mathtt{h}(N) : \mathtt{LL} \to \{\mathtt{h}(M) \sim \mathtt{h}(N)\}}\ (\textsc{THash}) \quad \frac{\Gamma \vdash M \sim N : \mathtt{LL} \to c}{\Gamma \vdash \mathtt{h}(M) \sim \mathtt{h}(N) : \mathtt{LL} \to c}\ (\textsc{THashL})$$

$$\frac{\begin{array}{c} M \downarrow\!\neq \bot \qquad N \downarrow\!\neq \bot \\ \mathrm{names}(M) \cup \mathrm{names}(N) \cup \mathrm{vars}(M) \cup \mathrm{vars}(N) \cup \mathrm{keys}(M) \cup \mathrm{keys}(N) \subseteq \\ \mathrm{dom}(\Gamma) \cup \mathrm{keys}(\Gamma) \cup \mathcal{FN} \cup \mathcal{FK} \end{array}}{\Gamma \vdash M \sim N : \mathtt{HL} \to \emptyset}\ (\textsc{THigh})$$

$$\frac{\Gamma \vdash M \sim N : T' \to c \qquad T' <: T}{\Gamma \vdash M \sim N : T \to c}\ (\textsc{TSub}) \quad \frac{\Gamma \vdash M \sim N : T \to c}{\Gamma \vdash M \sim N : T \vee T' \to c}\ (\textsc{TOr})$$

$$\frac{\begin{array}{l} \Gamma(m) = \tau_m^{l,1} \quad \text{or} \quad m \in \mathcal{FN} \cup \mathcal{C} \ \wedge\ l = \mathtt{LL} \\ \Gamma(n) = \tau_n^{l',1} \quad \text{or} \quad n \in \mathcal{FN} \cup \mathcal{C} \ \wedge\ l' = \mathtt{LL} \end{array}}{\Gamma \vdash m \sim n : [\![\tau_m^{l,1}\,;\, \tau_n^{l',1}]\!] \to \emptyset}\ (\textsc{TLR}^1) \quad \frac{\Gamma(m) = \tau_m^{l,\infty} \qquad \Gamma(n) = \tau_n^{l',\infty}}{\Gamma \vdash m \sim n : [\![\tau_m^{l,\infty}\,;\, \tau_n^{l',\infty}]\!] \to \emptyset}\ (\textsc{TLR}^\infty)$$

$$\frac{\Gamma \vdash M \sim N : [\![\tau_m^{l,a}\,;\, \tau_n^{l,a}]\!] \to c \qquad l \in \{\mathtt{HL}, \mathtt{HH}\}}{\Gamma \vdash M \sim N : l \to c}\ (\textsc{TLR'}) \quad \frac{\Gamma \vdash M \sim N : [\![\tau_n^{\mathtt{LL},a}\,;\, \tau_n^{\mathtt{LL},a}]\!] \to c}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}\ (\textsc{TLRL'})$$

$$\frac{\Gamma \vdash x \sim x : [\![\tau_m^{l,1}\,;\, \tau_n^{l',1}]\!] \to \emptyset \qquad \Gamma \vdash y \sim y : [\![\tau_{m'}^{l'',1}\,;\, \tau_{n'}^{l''',1}]\!] \to \emptyset}{\Gamma \vdash x \sim y : [\![\tau_m^{l,1}\,;\, \tau_{n'}^{l''',1}]\!] \to \emptyset}\ (\textsc{TLRVar})$$

Fig. 15. Rules for Messages

$$\frac{}{T <: T}\ (\textsc{SRefl}) \quad \frac{}{T <: \mathtt{HL}}\ (\textsc{SHigh}) \quad \frac{T <: T' \qquad T' <: T''}{T <: T''}\ (\textsc{STrans})$$

$$\frac{}{\mathtt{LL} * \mathtt{LL} <: \mathtt{LL}}\ (\textsc{SPairL}) \quad \frac{T_1 <: T_1' \qquad T_2 <: T_2'}{T_1 * T_2 <: T_1' * T_2'}\ (\textsc{SPair})$$

$$\frac{}{\mathtt{HH} * T <: \mathtt{HH}}\ (\textsc{SPairS}) \quad \frac{}{T * \mathtt{HH} <: \mathtt{HH}}\ (\textsc{SPairS'})$$

$$\frac{}{\mathrm{key}^l(T) <: l}\ (\textsc{SKey}) \quad \frac{}{\mathrm{eqkey}^l(T) <: \mathrm{key}^l(T)}\ (\textsc{SEqKey}) \quad \frac{}{\mathrm{seskey}^{l,a}(T) <: \mathrm{eqkey}^l(T)}\ (\textsc{SSesKey})$$

$$\frac{T <: \mathrm{eqkey}^l(T')}{\mathrm{pkey}(T) <: \mathtt{LL}}\ (\textsc{SPubKey}) \quad \frac{T <: \mathrm{eqkey}^l(T')}{\mathrm{vkey}(T) <: \mathtt{LL}}\ (\textsc{SVKey})$$

$$\frac{T <: T'}{(T)_{T''} <: (T')_{T''}}\ (\textsc{SEnc}) \quad \frac{T <: T'}{\{T\}_{T''} <: \{T'\}_{T''}}\ (\textsc{SAEnc})$$

Fig. 16. Subtyping Rules

Definition 4 (Constraint). *A constraint is defined as a couple of messages, separated by the symbol $\sim$:*

$$u \sim v$$

We will consider sets of constraints, which we usually denote c. We will also consider couples (c, Γ) composed of such a set, and a typing environment Γ. Finally we will denote sets of such tuples C, and call them *constraint sets*.

Definition 5 (Compatible environments). *We say that two typing environments Γ, Γ' are compatible if they are equal on the intersection of their domains,* i.e. *if*

$$\forall x \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma').\ \Gamma(x) = \Gamma'(x)$$

Definition 6 (Union of environments). *Let Γ, Γ' be two compatible environments. Their union $\Gamma \cup \Gamma'$ is defined by*

- $\mathrm{dom}(\Gamma \cup \Gamma') = \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Gamma')$
- $\forall x \in \mathrm{dom}(\Gamma).\quad (\Gamma \cup \Gamma')(x) = \Gamma(x)$
- $\forall x \in \mathrm{dom}(\Gamma').\quad (\Gamma \cup \Gamma')(x) = \Gamma'(x)$

Note that this function is well defined since Γ and Γ' are assumed to be compatible.

Definition 7 (Operations on constraint sets). *We define two operations on constraints.*

- *the product union of constraint sets:*

$$\begin{aligned} C \cup_\times C' := \{&(c \cup c', \Gamma \cup \Gamma') \mid \\ &(c, \Gamma) \in C \ \wedge\ (c', \Gamma') \in C' \ \wedge\ \Gamma, \Gamma' \text{ are compatible}\} \end{aligned}$$

- *the addition of a set of constraints c' to all elements of a constraint set C:*

$$\begin{aligned} C \cup_\forall c' :=& C \cup_\times \{(c', \emptyset)\} \\ =& \{(c \cup c', \Gamma) \mid (c, \Gamma) \in C\} \end{aligned}$$

$$\frac{\Gamma \vdash \diamond \qquad \text{branches}(\Gamma) = \{\Gamma\}}{\Gamma \vdash \mathtt{0} \sim \mathtt{0} \to (\emptyset, \Gamma)}\ (\textsc{PZero})$$

$$\frac{\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c}{\Gamma \vdash \mathtt{out}(M).P \sim \mathtt{out}(N).Q \to C \cup_\forall c}\ (\textsc{POut}) \qquad \frac{\Gamma, x : \mathtt{LL} \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{in}(x).P \sim \mathtt{in}(x).Q \to C}\ (\textsc{PIn})$$

$$\frac{\Gamma, n : \tau_n^{l,a} \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{new}\ n : \tau_n^{l,a}.P \sim \mathtt{new}\ n : \tau_n^{l,a}.Q \to C}\ (\textsc{PNew})$$

$$\frac{\Gamma, (k,k) : \text{seskey}^{l,a}(T) \vdash P \sim Q \to C}{\Gamma \vdash \mathtt{new}\ k : \text{seskey}^{l,a}(T).P \sim \mathtt{new}\ k : \text{seskey}^{l,a}(T).Q \to C}\ (\textsc{PNewKey})$$

$$\frac{\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C'}{\Gamma \vdash P \mid P' \sim Q \mid Q' \to C \cup_\times C'}\ (\textsc{ppar}) \qquad \frac{\Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma, x : T' \vdash P \sim Q \to C'}{\Gamma, x : T \vee T' \vdash P \sim Q \to C \cup C'}\ (\textsc{por})$$

$$\frac{\Gamma \vdash_d t \sim t' : T \qquad \Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C'}{\Gamma \vdash \mathtt{let}\ x = t\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = t'\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to C \cup C'}\ (\textsc{PLet})$$

$$\frac{\begin{array}{c}\Gamma(y) = \mathtt{LL} \qquad \Gamma(k_1,k_2) <: \text{key}^{\mathtt{HH}}(T) \\ \Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C' \\ (\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_1,k_3) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P \sim Q' \to C_{k_3}) \\ (\forall T'.\forall k_3 \neq k_1.\ \Gamma(k_3,k_2) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P' \sim Q \to C'_{k_3})\end{array}}{\begin{array}{l}\Gamma \vdash \mathtt{let}\ x = \mathtt{dec}(y,k_1)\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = \mathtt{dec}(y,k_2)\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to \\ \qquad C \cup C' \cup (\bigcup_{k_3} C_{k_3}) \cup (\bigcup_{k_3} {C'}_{k_3})\end{array}}\ (\textsc{PLetDec})$$

$$\frac{\begin{array}{c}\Gamma(y) = \mathtt{LL} \qquad \Gamma(k,k) <: \text{key}^{\mathtt{HH}}(T) \qquad \Gamma, x : T \vdash P \sim Q \to C \\ \Gamma, x : \mathtt{LL} \vdash P \sim Q \to C' \qquad \Gamma \vdash P' \sim Q' \to C'' \\ (\forall T'.\forall k_3 \neq k.\ \Gamma(k,k_3) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P \sim Q' \to C_{k_3}) \\ (\forall T'.\forall k_3 \neq k.\ \Gamma(k_3,k) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P' \sim Q \to C'_{k_3})\end{array}}{\begin{array}{l}\Gamma \vdash \mathtt{let}\ x = \mathtt{adec}(y,k)\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = \mathtt{adec}(y,k)\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to \\ \qquad C \cup C' \cup C'' \cup (\bigcup_{k_3} C_{k_3}) \cup (\bigcup_{k_3} {C'}_{k_3})\end{array}}\ (\textsc{PLetAdecSame})$$

$$\frac{\begin{array}{c}k_1 \neq k_2 \qquad \Gamma(y) = \mathtt{LL} \qquad \Gamma(k_1,k_2) <: \text{key}^{\mathtt{HH}}(T) \\ \Gamma, x : T \vdash P \sim Q \to C \qquad \Gamma, x : \mathtt{LL} \vdash P \sim Q' \to C' \\ \Gamma, x : \mathtt{LL} \vdash P' \sim Q \to C'' \qquad \Gamma \vdash P' \sim Q' \to C''' \\ (\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_1,k_3) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P \sim Q' \to C_{k_3}) \\ (\forall T'.\forall k_3 \neq k_1.\ \Gamma(k_3,k_2) <: \text{key}^{\mathtt{HH}}(T') \Rightarrow \Gamma, x : T' \vdash P' \sim Q \to C'_{k_3})\end{array}}{\begin{array}{l}\Gamma \vdash \mathtt{let}\ x = \mathtt{adec}(y,k_1)\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = \mathtt{adec}(y,k_2)\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to \\ \qquad C \cup C' \cup C'' \cup C''' \cup (\bigcup_{k_3} C_{k_3}) \cup (\bigcup_{k_3} {C'}_{k_3})\end{array}}\ (\textsc{PLetAdecDiff})$$

$$\frac{\Gamma(y) = [\![\tau_n^{l,a}\,;\,\tau_m^{l',a}]\!] \vee \Gamma(y) <: \text{key}^{l}(T) \qquad \Gamma \vdash P' \sim Q' \to C'}{\Gamma \vdash \mathtt{let}\ x = d(y)\ \mathtt{in}\ P\ \mathtt{else}\ P' \sim \mathtt{let}\ x = d(y)\ \mathtt{in}\ Q\ \mathtt{else}\ Q' \to C'}\ (\textsc{PLetLRK})$$

Fig. 17. Rules for processes (1)

$$\frac{\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C' \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c \qquad \Gamma \vdash M' \sim N' : \mathtt{LL} \to c'}{\Gamma \vdash \mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = N'\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to \big(C \cup C'\big) \cup_\forall (c \cup c')}\ (\textsc{PIfL})$$

$$\frac{\begin{array}{c}\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,1}\,;\,\tau_n^{l',1}]\!] \to \emptyset \qquad \Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',1}\,;\,\tau_{n'}^{l''',1}]\!] \to \emptyset \\ b = (\tau_m^{l,1} \stackrel{?}{=} \tau_{m'}^{l'',1}) \qquad b' = (\tau_n^{l',1} \stackrel{?}{=} \tau_{n'}^{l''',1}) \qquad \Gamma \vdash P_b \sim Q_{b'} \to C\end{array}}{\Gamma \vdash \mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P_\top\ \mathtt{else}\ P_\bot \sim \mathtt{if}\ N_1 = N_2\ \mathtt{then}\ Q_\top\ \mathtt{else}\ Q_\bot \to C}\ (\textsc{PIfLR})$$

$$\frac{\Gamma \vdash P' \sim Q' \to C' \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c \qquad \Gamma \vdash M' \sim N' : \mathtt{HH} \to c'}{\Gamma \vdash \mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = N'\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C'}\ (\textsc{PIfS})$$

$$\frac{\begin{array}{c}\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,\infty}\,;\,\tau_n^{l',\infty}]\!] \to \emptyset \qquad \Gamma \vdash M_2 \sim N_2 : [\![\tau_m^{l,\infty}\,;\,\tau_n^{l',\infty}]\!] \to \emptyset \\ \Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C'\end{array}}{\Gamma \vdash \mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N_1 = N_2\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C \cup C'}\ (\textsc{PIfLR*})$$

$$\frac{\begin{array}{c}\Gamma \vdash P \sim Q \to C \qquad \Gamma \vdash P' \sim Q' \to C' \\ \Gamma \vdash M \sim N : \mathtt{LL} \to c \qquad \Gamma \vdash t \sim t : \mathtt{LL} \to c' \qquad t \in \mathcal{K} \cup \mathcal{N} \cup \mathcal{C}\end{array}}{\Gamma \vdash \mathtt{if}\ M = t\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = t\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C \cup C'}\ (\textsc{PIfP})$$

$$\frac{\Gamma \vdash P' \sim Q' \to C' \qquad \Gamma \vdash M \sim N : T * T' \to c \qquad \Gamma \vdash M' \sim N' : [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \to \emptyset}{\Gamma \vdash \mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = N'\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C'}\ (\textsc{PIfI})$$

$$\frac{\begin{array}{c}\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \to \emptyset \qquad \Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',a'}\,;\,\tau_{n'}^{l''',a'}]\!] \to \emptyset \\ \tau_m^{l,a} \neq \tau_{m'}^{l'',a'}, \tau_n^{l',a} \neq \tau_{n'}^{l''',a'} \qquad \Gamma \vdash P' \sim Q' \to C\end{array}}{\Gamma \vdash \mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N_1 = N_2\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C}\ (\textsc{PIfLR'*})$$

$$\frac{\Gamma \vdash P \sim Q \to C_1 \qquad \Gamma \vdash P \sim Q' \to C_2 \qquad \Gamma \vdash P' \sim Q \to C_3 \qquad \Gamma \vdash P' \sim Q' \to C_4}{\Gamma \vdash \mathtt{if}\ M = M'\ \mathtt{then}\ P\ \mathtt{else}\ P' \sim \mathtt{if}\ N = N'\ \mathtt{then}\ Q\ \mathtt{else}\ Q' \to C_1 \cup C_2 \cup C_3 \cup C_4}\ (\textsc{PIfAll})$$

Fig. 18. Rules for processes (2)

$$\frac{\Gamma(k,k) <: \mathrm{key}^{\mathtt{LL}}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{dec}(x,k) \sim \mathtt{dec}(x,k) : \mathtt{LL}}\ (\textsc{DDecL})$$

$$\frac{\Gamma(k,k) = \mathrm{seskey}^{l,a}(T') \qquad \Gamma(x) = (T)_{\mathrm{seskey}^{l,a}(T')}}{\Gamma \vdash_d \mathtt{dec}(x,k) \sim \mathtt{dec}(x,k) : T}\ (\textsc{DDecT})$$

$$\frac{\Gamma(y) = \mathrm{seskey}^{l,a}(T') \qquad \Gamma(x) = (T)_{\mathrm{seskey}^{l,a}(T')}}{\Gamma \vdash_d \mathtt{dec}(x,y) \sim \mathtt{dec}(x,y) : T}\ (\textsc{DDecT'})$$

$$\frac{\Gamma(y) = \mathrm{seskey}^{\mathtt{HH},a}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{dec}(x,y) \sim \mathtt{dec}(x,y) : T}\ (\textsc{DDecH'})$$

$$\frac{(\Gamma(y) = \mathrm{seskey}^{\mathtt{LL},a}(T) \ \vee \ \Gamma(y) = \mathtt{LL}) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{dec}(x,y) \sim \mathtt{dec}(x,y) : \mathtt{LL}}\ (\textsc{DDecL'})$$

$$\frac{\Gamma(k,k) <: \mathrm{key}^{\mathtt{LL}}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{adec}(x,k) \sim \mathtt{adec}(x,k) : \mathtt{LL}}\ (\textsc{DAdecL'})$$

$$\frac{\Gamma(y) = \mathrm{seskey}^{\mathtt{HH},a}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{adec}(x,y) \sim \mathtt{adec}(x,y) : T \ \vee \ \mathtt{LL}}\ (\textsc{DAdecH'})$$

$$\frac{(\Gamma(y) = \mathrm{seskey}^{\mathtt{LL},a}(T) \ \vee \ \Gamma(y) = \mathtt{LL}) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{adec}(x,y) \sim \mathtt{adec}(x,y) : \mathtt{LL}}\ (\textsc{DAdecL'})$$

$$\frac{\Gamma(k,k) = \mathrm{seskey}^{l,a}(T') \qquad \Gamma(x) = \{T\}_{\mathrm{pkey}(\mathrm{seskey}^{l,a}(T'))}}{\Gamma \vdash_d \mathtt{adec}(x,k) \sim \mathtt{adec}(x,k) : T}\ (\textsc{DAdecT})$$

$$\frac{\Gamma(y) = \mathrm{seskey}^{l,a}(T') \qquad \Gamma(x) = \{T\}_{\mathrm{pkey}(\mathrm{seskey}^{l,a}(T'))}}{\Gamma \vdash_d \mathtt{adec}(x,y) \sim \mathtt{adec}(x,y) : T}\ (\textsc{DAdecT'})$$

$$\frac{\Gamma(k,k) <: \mathrm{key}^{\mathtt{HH}}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{checksign}(x, \mathtt{vk}(k)) \sim \mathtt{checksign}(x, \mathtt{vk}(k)) : T}\ (\textsc{DCheckH})$$

$$\frac{\Gamma(k,k) <: \mathrm{key}^{\mathtt{LL}}(T) \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{checksign}(x, \mathtt{vk}(k)) \sim \mathtt{checksign}(x, \mathtt{vk}(k)) : \mathtt{LL}}\ (\textsc{DCheckL})$$

$$\frac{\Gamma(y) = \mathrm{vkey}(T) \qquad T <: \mathrm{eqkey}^{\mathtt{HH}}(T') \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{checksign}(x,y) \sim \mathtt{checksign}(x,y) : T'}\ (\textsc{DCheckH'})$$

$$\frac{(\Gamma(y) = \mathrm{vkey}(T) \wedge T <: \mathrm{eqkey}^{\mathtt{LL}}(T')) \text{ or } \Gamma(y) = \mathtt{LL} \qquad \Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \mathtt{checksign}(x,y) \sim \mathtt{checksign}(x,y) : \mathtt{LL}}\ (\textsc{DCheckL'})$$

$$\frac{\Gamma(x) = T * T'}{\Gamma \vdash_d \pi_1(x) \sim \pi_1(x) : T}\ (\textsc{DFst}) \qquad \frac{\Gamma(x) = T * T'}{\Gamma \vdash_d \pi_2(x) \sim \pi_2(x) : T'}\ (\textsc{DSnd})$$

$$\frac{\Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \pi_1(x) \sim \pi_1(x) : \mathtt{LL}}\ (\textsc{DFstL}) \qquad \frac{\Gamma(x) = \mathtt{LL}}{\Gamma \vdash_d \pi_2(x) \sim \pi_2(x) : \mathtt{LL}}\ (\textsc{DSndL})$$

Fig. 19. Destructor Rules

Definition 8. *For any typing environment Γ, we denote by $\Gamma_{\mathcal{X}}$ its restriction to variables, and by $\Gamma_{\mathcal{N},\mathcal{K}}$ its restriction to names and key couples.*

Definition 9 (Well-typed substitutions). *Let Γ be a typing environment, θ, θ' two substitutions, and c a set of constraints. We say that θ, θ' are well-typed in Γ, and write $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \theta \sim \theta' : \Gamma_{\mathcal{X}} \to c$, if they are ground and*

- $\mathrm{dom}(\theta) = \mathrm{dom}(\theta') = \mathrm{dom}(\Gamma_{\mathcal{X}})$,
- *and*

$$\forall x \in \mathrm{dom}(\Gamma_{\mathcal{X}}),\ \Gamma_{\mathcal{N},\mathcal{K}} \vdash \theta(x) \sim \theta'(x) : \Gamma(x) \to c_x$$

for some c_x such that $c = \bigcup_{x \in \mathrm{dom}(\Gamma_{\mathcal{X}})} c_x$.

Definition 10 (LL substitutions). *Let Γ be an environment, ϕ, ϕ' two substitutions and c a set of constraints. We say that ϕ, ϕ' have type $\mathtt{LL}$ in Γ with constraint c, and write $\Gamma \vdash \phi \sim \phi' : \mathtt{LL} \to c$ if*

- $\mathrm{dom}(\phi) = \mathrm{dom}(\phi')$;
- *for all $x \in \mathrm{dom}(\phi)$ there exists c_x such that $\Gamma \vdash \phi(x) \sim \phi'(x) : \mathtt{LL} \to c_x$ and $c = \bigcup_{x \in \mathrm{dom}(\phi)} c_x$.*

Definition 11 (Frames associated to a set of constraints). *If c is a set of constraints, let $\phi_\ell(c)$ and $\phi_r(c)$ be the frames composed of the terms respectively on the left and on the right of the $\sim$ symbol in the constraints of c (in the same order).*

Definition 12 (Instantiation of constraints). *If c is a set of constraints, and σ, σ' are two substitutions, let $[\![c]\!]_{\sigma,\sigma'}$ be the instantiation of c by σ on the left and σ' on the right,* i.e.

$$[\![c]\!]_{\sigma,\sigma'} = \{M\sigma \sim N\sigma' \mid M \sim N \in c\}.$$

Similarly we write for a constraint set C

$$[\![C]\!]_{\sigma,\sigma'} = \{([\![c]\!]_{\sigma,\sigma'}, \Gamma) \mid (c, \Gamma) \in C\}.$$

Definition 13 (Frames associated to environments). *If Γ is a typing environment, we denote $\phi^{\Gamma}_{\mathtt{LL}}$ the frame containing all the keys k such that $\Gamma(k,k) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, all the public keys $\mathtt{pk}(k)$ and $\mathtt{vk}(k)$ for $k \in \mathrm{keys}(\Gamma)$, and all the nonces n such that $\Gamma(n) = \tau_n^{\mathtt{LL},a}$ (for $a \in \{\infty, 1\}$).*

Definition 14 (Branches of a type). *If T is a type, we write $\mathrm{branches}(T)$ the set of all types T' such that T' is not a union type, and either*

- $T = T'$;
- *or there exist types $T_1,\ldots,T_k,T'_1,\ldots,T'_{k'}$ such that*

$$T = T_1 \vee \ldots \vee T_k \vee T \vee T'_1 \vee \ldots \vee T'_{k'}$$

Definition 15 (Branches of an environment). *For a typing environment Γ, we write $\mathrm{branches}(\Gamma)$ the sets of all environments Γ' such that*

- $\mathrm{dom}(\Gamma') = \mathrm{dom}(\Gamma)$
- $\forall x \in \mathrm{dom}(\Gamma).\quad \Gamma'(x) \in \mathrm{branches}(\Gamma(x))$.

Definition 16 (Consistency). *We say that c is consistent in a typing environment Γ, if for all subsets $c' \subseteq c$ and $\Gamma' \subseteq \Gamma$ such that $\Gamma'_{\mathcal{N},\mathcal{K}} = \Gamma_{\mathcal{N},\mathcal{K}}$ and $\mathrm{vars}(c') \subseteq \mathrm{dom}(\Gamma')$, for all ground substitutions σ, σ', if there exists a constraint c_σ such that $(\Gamma')_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : (\Gamma')_{\mathcal{X}} \to c_\sigma$ and $c_\sigma \subseteq [\![c']\!]_{\sigma,\sigma'}$, then the frames $\phi^{\Gamma}_{\mathtt{LL}} \cup \phi_\ell(c')\sigma$ and $\phi^{\Gamma}_{\mathtt{LL}} \cup \phi_r(c')\sigma'$ are statically equivalent.*

We say that (c, Γ) is consistent if c is consistent in Γ.

We say that a constraint set C is consistent if each element $(c, \Gamma) \in C$ is consistent.

B Proofs

In this section, we provide the detailed proofs to all of our theorems.

Unless specified otherwise, the environments Γ considered in the lemmas are implicitly assumed to be well-formed.

B.1 General results and soundness

In this subsection, we prove soundness for non replicated processes, as well as several results regarding the type system that this proof uses.

Lemma 3 (Subtyping properties). *The following properties of subtyping hold:*

1. $\forall T.\quad \mathtt{HL} <: T \implies T = \mathtt{HL}$
2. $\forall T.\quad \mathtt{LL} <: T \implies T = \mathtt{LL} \lor T = \mathtt{HL}$
3. $\forall T.\quad \mathtt{HH} <: T \implies T = \mathtt{HH} \lor T = \mathtt{HL}$
4. $\forall T_1, T_2, T_3.\quad T_1 * T_2 <: T_3 \implies T_3 = \mathtt{LL} \lor T_3 = \mathtt{HL} \lor T_3 = \mathtt{HH} \lor (\exists T_4, T_5.\quad T_3 = T_4 * T_5)$ i.e. T_3 *is* LL, HL, HH *or a pair type.*
5. $\forall T, T_1, T_2.\quad T <: T_1 * T_2 \implies (\exists T'_1, T'_2.\quad T = T'_1 * T'_2 \land T'_1 <: T_1 \land T'_2 <: T_2)$
6. $\forall T_1, T_2.\quad T_1 * T_2 <: \mathtt{LL} \implies T_1 <: \mathtt{LL} \land T_2 <: \mathtt{LL}$
7. $\forall T_1, T_2.\quad T_1 * T_2 <: \mathtt{HH} \implies T_1 <: \mathtt{HH} \lor T_2 <: \mathtt{HH}$
8. $\forall T_1, T_2, T_3.\quad T_1 <: (T_2)_{T_3} \implies (\exists T_4 <: T_2.\quad T_1 = (T_4)_{T_3})$
9. $\forall T_1, T_2, T_3.\quad T_1 <: \{T_2\}_{T_3} \implies (\exists T_4 <: T_2.\quad T_1 = \{T_4\}_{T_3})$
10. $\forall T_1, T_2, T_3.\quad (T_1)_{T_2} <: T_3 \implies T_3 = \mathtt{HL} \lor T_3 = \mathtt{LL} \lor (\exists T_4. T_1 <: T_4 \land T_3 = (T_4)_{T_2})$
11. $\forall T_1, T_2, T_3.\quad \{T_1\}_{T_2} <: T_3 \implies T_3 = \mathtt{HL} \lor T_3 = \mathtt{LL} \lor (\exists T_4. T_1 <: T_4 \land T_3 = \{T_4\}_{T_2})$
12. $\forall T_1, T_2.\quad (T_1)_{T_2} <: \mathtt{LL} \implies T_1 <: \mathtt{LL} \land (T_2 = \mathtt{LL} \lor (\exists T_3. T_2 <: \mathrm{key}^{\mathtt{LL}}(T_3)))$
13. $\forall T_1, T_2.\quad \{T_1\}_{T_2} <: \mathtt{LL} \implies T_1 <: \mathtt{LL} \land (T_2 = \mathtt{LL} \lor (\exists T_3, T_4, l. T_2 = \mathrm{pkey}(T_3) \land T_3 <: \mathrm{eqkey}^l(T_4)))$
14. $\forall T, m, n, l, l'.\quad T <: [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \implies T = [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!]$
15. $\forall T, m, n, l, l'.\quad [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] <: T \implies T = \mathtt{HL} \lor T = [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!]$
16. $\forall T_1, T_2.\quad T_1 <: T_2 \implies$ *neither* T_1 *nor* T_2 *are union types unless* $T_2 = \mathtt{HL}$ *or* $T_1 = T_2$.
17. $\forall T, l, T'.\quad T <: \mathrm{key}^l(T') \implies T = \mathrm{key}^l(T') \lor T = \mathrm{eqkey}^l(T') \lor \exists a. T = \mathrm{seskey}^{l,a}(T')$.
18. $\forall T, l, T'.\quad T <: \mathrm{eqkey}^l(T') \implies T = \mathrm{eqkey}^l(T') \lor T = \mathrm{seskey}^{l,a}(T')$.
19. $\forall T, l, a, T'.\quad T <: \mathrm{seskey}^{l,a}(T') \implies T = \mathrm{seskey}^{l,a}(T')$.
20. $\forall T, T'.\quad T <: \mathrm{pkey}(T') \implies T = \mathrm{pkey}(T')$.
21. $\forall T, T'.\quad T <: \mathrm{vkey}(T') \implies T = \mathrm{vkey}(T')$.
22. $\forall T_1, T_2, l.\quad \mathrm{key}^l(T_1) <: T_2 \implies T_2 = l \lor T_2 = \mathtt{HL} \lor T_2 = \mathrm{key}^l(T)$.
23. $\forall T_1, T_2, l.\quad \mathrm{eqkey}^l(T_1) <: T_2 \implies T_2 = l \lor T_2 = \mathtt{HL} \lor T_2 = \mathrm{key}^l(T) \lor T_2 = \mathrm{eqkey}^l(T)$.
24. $\forall T_1, T_2, l.\quad \mathrm{seskey}^{l,a}(T_1) <: T_2 \implies T_2 = l \lor T_2 = \mathtt{HL} \lor T_2 = \mathrm{key}^l(T) \lor T_2 = \mathrm{eqkey}^l(T) \lor T_2 = \mathrm{seskey}^{l,a}(T)$.
25. $\forall T_1, T_2.\quad \mathrm{pkey}(T_1) <: T_2 \implies T_2 = \mathtt{HL} \lor T_2 = \mathtt{LL} \lor T_2 = \mathrm{pkey}(T_1)$.
26. $\forall T_1.\quad \mathrm{pkey}(T_1) <: \mathtt{LL} \implies \exists T_3, l.\, T_1 <: \mathrm{eqkey}^l(T_3)$.
27. $\forall T_1, T_2.\quad \mathrm{vkey}(T_1) <: T_2 \implies T_2 = \mathtt{HL} \lor T_2 = \mathtt{LL} \lor T_2 = \mathrm{vkey}(T_1)$.
28. $\forall T_1.\quad \mathrm{vkey}(T_1) <: \mathtt{LL} \implies \exists T_3, l.\, T_1 <: \mathrm{eqkey}^l(T_3)$.
29. $\forall T.\quad T <: \mathtt{LL} \implies T$ *is a pair type* $\lor (\exists T', T''.\quad T = (T')_{T''}) \lor (\exists T', T''.\quad T = \{T'\}_{T''}) \lor (\exists T'.\quad T <: \mathrm{key}^{\mathtt{LL}}(T')) \lor (\exists l, T'.\quad T = \mathrm{pkey}(T')) \lor (\exists l, T'.\quad T = \mathrm{vkey}(T')) \lor T = \mathtt{LL}$.
30. $\forall T.\quad T <: \mathtt{HH} \implies T$ *is a pair type* $\lor (\exists T'.\quad T <: \mathrm{key}^{\mathtt{HH}}(T')) \lor T = \mathtt{HH}$.

Proof. All these properties have simple proofs by induction on the subtyping derivation.

Lemma 4 (Terms of type $T \vee T'$). *For all Γ, T, T', for all ground terms t, t', for all c, if*

$$\Gamma \vdash t \sim t' : T \vee T' \to c$$

then

$$\Gamma \vdash t \sim t' : T \to c \quad \textit{or} \quad \Gamma \vdash t \sim t' : T' \to c$$

Proof. We prove this property by induction on the derivation of $\Gamma \vdash t \sim t' : T \vee T' \to c$.

The last rule of the derivation cannot be TNONCE, TNONCEL, TCSTFN, TPUBKEY, TPUBKEYL, TVKEY, TVKEYL, TPAIR, TKEY, TPAIR, TENC, TENCH, TENCL, TAENC, TAENCH, TAENCL, TSIGNH, TSIGNL, THASH, THASHL, THIGH, TLR^1, TLR$^\infty$, TLRVAR, TLR', or TLRL' since the type in their conclusion cannot be $T \vee T'$. It cannot be TVAR since t, t' are ground.

In the TSUB case we know that $\Gamma \vdash t \sim t' : T'' \to c$ (with a shorter derivation) for some $T'' <: T \vee T'$; thus, by Lemma 3, $T'' = T \vee T'$, and the claim holds by the induction hypothesis.

Finally in the TOR case, the premise of the rule directly proves the claim.

Lemma 5 (Terms and branch types). *For all Γ, T, c, for all ground terms t, t', if*

$$\Gamma \vdash t \sim t' : T \to c$$

then there exists $T' \in \mathrm{branches}(T)$ such that

$$\Gamma \vdash t \sim t' : T' \to c$$

Proof. This property is a corollary of Lemma 4. We indeed prove it by successively applying this lemma to $\Gamma \vdash t \sim t' : T \to c$ until T is not a union type.

Lemma 6 (Substitutions type in a branch). *For all Γ, c, for all ground substitutions σ, σ', if*

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma_{\mathcal{X}} \to c$$

then there exists $\Gamma' \in \mathrm{branches}(\Gamma)$ such that

$$\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c$$

Proof. This property follows from Lemma 5. Indeed, by definition, $c = \bigcup_{x \in \mathrm{dom}(\Gamma_{\mathcal{X}})} c_x$ for some c_x such that for all $x \in \mathrm{dom}(\Gamma_{\mathcal{X}})(= \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma'))$,

$$\Gamma \vdash \sigma(x) \sim \sigma'(x) : \Gamma(x) \to c_x$$

Hence by applying Lemma 5 we obtain a type $T_x \in \mathrm{branches}(\Gamma(x))$ such that

$$\Gamma \vdash \sigma(x) \sim \sigma'(x) : T_x \to c_x$$

Thus if we denote Γ'' by $\forall x \in \mathrm{dom}(\Gamma_{\mathcal{X}}).\Gamma''(x) = T_x$, and $\Gamma' = \Gamma_{\mathcal{N},\mathcal{K}} \cup \Gamma''$, we have $\Gamma' \in \mathrm{branches}(\Gamma)$ and $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c$.

Lemma 7 (Typing terms in branches). *For all Γ, T, c, for all terms t, t', for all $\Gamma' \in \mathrm{branches}(\Gamma)$, if $\Gamma \vdash t \sim t' : T \to c$ then $\Gamma' \vdash t \sim t' : T \to c$.*

Corollary: in that case, there exists $T' \in \mathrm{branches}(T)$ such that $\Gamma' \vdash t \sim t' : T' \to c$.

Proof. We prove this property by induction on the derivation of $\Gamma \vdash t \sim t' : T \to c$. In most cases for the last rule applied, $\Gamma(x)$ is not directly involved in the premises, for any variable x. Rather, Γ appears only in other typing judgements, or is used in $\Gamma(k, k')$ or $\Gamma(n)$ for some keys k, k' or nonce n, and keys or nonces cannot have union types. Hence, since the typing rules for terms do not change Γ, the claim directly follows from the induction hypothesis. For instance in the TPAIR case, we have $t = \langle t_1, t_2 \rangle$, $t' = \langle t'_1, t'_2 \rangle$, $T = T_1 * T_2$, $c = c_1 \cup c_2$, $\Gamma \vdash t_1 \sim t'_1 : T_1 \to c_1$, and $\Gamma \vdash t_2 \sim t'_2 : T_2 \to c_2$. Thus by the induction hypothesis, $\Gamma' \vdash t_1 \sim t'_1 : T_1 \to c_1$, and $\Gamma' \vdash t_2 \sim t'_2 : T_2 \to c_2$; and therefore by rule TPAIR, $\Gamma' \vdash t \sim t' : T \to c$. The cases of rules TPUBKEY, TVKEY, TKEY, TENC, TENCH, TENCL, TAENC, TAENCH, TAENCL, THASHL, TSIGNH, TSIGNL, TLR', TLRL', TLRVAR, TSUB, TOR are similar.

The cases of rules TNONCE, TNONCEL, TCSTFN, TPUBKEYL, TVKEYL, THASH, THIGH, TLR[1], and TLR$^\infty$ are immediate since these rules use neither Γ nor another typing judgement in their premise.

Finally, in the TVAR case, $t = t' = x$ for some variable x such that $\Gamma(x) = T$, and $c = \emptyset$. Rule TVAR also proves that $\Gamma' \vdash x \sim x : \Gamma'(x) \to \emptyset$. Since $\Gamma'(x) \in \mathrm{branches}(\Gamma(x))$, by applying rule TOR as many times as necessary, we have $\Gamma' \vdash x \sim x : \Gamma(x) \to \emptyset$, *i.e.* $\Gamma' \vdash x \sim x : T \to \emptyset$, which proves the claim.

The corollary then follows, again by induction on the typing derivation. If T is not a union type, $\mathrm{branches}(T) = \{T\}$ and the claim is directly the previous property. Otherwise, the last rule applied in the typing derivation can only be TVAR, TSUB, or TOR. The TSUB case follows trivially from the induction hypothesis; since T is a union type, it is its own only subtype. In the TVAR case, $t = t' = x$ for some variable x such that $\Gamma(x) = T$. Hence, by definition, $\Gamma'(x) \in \mathrm{branches}(T)$, and by rule TVAR we have $\Gamma' \vdash t \sim t' : \Gamma'(x) \to c$. Finally, in the TOR case, we have $T = T_1 \vee T_2$ for some T_1, T_2 such that $\Gamma \vdash t \sim t' : T_1 \to c$. By the induction hypothesis, there exists $T'_1 \in \mathrm{branches}(T_1)$ such that $\Gamma' \vdash t \sim t' : T'_1 \to c$. Since, by definition, $\mathrm{branches}(T_1) \subseteq \mathrm{branches}(T_1 \vee T_2)$, this proves the claim.

Lemma 8 (Typing destructors in branches). *For all Γ, T, t, t', x, for all $\Gamma' \in \mathrm{branches}(\Gamma)$, if $\Gamma \vdash_d t \sim t' : T$ then $\Gamma' \vdash_d t \sim t' : T$.*

Proof. This property is immediate by examining the typing rules for destructors. Indeed, Γ and Γ' only differ on variables, and the rules for destructors only involve $\Gamma(x)$ for $x \in \mathcal{X}$ in conditions of the form $\Gamma(x) = T$ for some type T which is not a union type.

Hence in these cases $\Gamma'(x)$ is also T, and the same rule can be applied to Γ' to prove the claim.

Lemma 9 (Typing processes in branches). *For all Γ, C, for all processes P, Q, for all $\Gamma' \in \mathrm{branches}(\Gamma)$, if $\Gamma \vdash P \sim Q \to C$ then there exists $C' \subseteq C$ such that $\Gamma' \vdash P \sim Q \to C'$.*

Proof. We prove this lemma by induction on the derivation of $\Gamma \vdash P \sim Q \to C$. In all the cases for the last rule applied in this derivation, we can show that the conditions of this rule still hold in Γ' (instead of Γ) using

- Lemma 7 for the conditions of the form $\Gamma \vdash M \sim N : T \to c$;
- Lemma 8 for the conditions of the form $\Gamma \vdash_d d(y) \sim d'(y) : T$;
- the fact that if $\Gamma(x)$ is not a union type, then $\Gamma'(x) = \Gamma(x)$, for conditions such as "$\Gamma(x) = \mathtt{LL}$", "$\Gamma(x) = [\![\tau_m^{l,a}\,;\, \tau_n^{l',a}]\!]$" or "$\Gamma(x) <: \mathrm{key}^l(T)$" (in the PLETLRK case);
- the induction hypothesis for the conditions of the form $\Gamma \vdash P' \sim Q' \to C''$. In this case, the induction hypothesis produces a $C''' \subseteq C''$, which can then be used to show $C' \subseteq C$, since C' and C are usually respectively C''' and C'' with some terms added.

We detail here the cases of rules POUT, PPAR, and POR. The other cases are similar, as explained above.

If the last rule is POUT, then we have $P = \mathtt{out}(M).P'$, $Q = \mathtt{out}(N).Q'$, $C = C'' \cup_\forall c$ for some P', Q', M, N, C'', c, such that $\Gamma \vdash P' \sim Q' \to C''$ and $\Gamma \vdash M \sim N : \mathtt{LL} \to c$. Hence by Lemma 7,

$\Gamma' \vdash M \sim N : \mathtt{LL} \to c$, and by the induction hypothesis applied to P', Q', $\Gamma' \vdash P' \sim Q' \to C'''$ for some C''' such that $C''' \subseteq C''$. Therefore by rule POUT, $\Gamma' \vdash P \sim Q \to C'''\cup_\forall c$, and since $C'''\cup_\forall c \subseteq C''\cup_\forall c (= C)$, this proves the claim.

If the last rule is PPAR, then we have $P = P_1 \mid P_2$, $Q = Q_1 \mid Q_2$, $C = C_1 \cup_\times C_2$ for some P_1, P_2, Q_1, Q_2, C_1, C_2 such that $\Gamma \vdash P_1 \sim Q_1 \to C_1$ and $\Gamma \vdash P_2 \sim Q_2 \to C_2$. Thus by applying the induction hypothesis twice, we have $\Gamma' \vdash P_1 \sim Q_1 \to C_1'$ and $\Gamma' \vdash P_2 \sim Q_2 \to C_2'$ with $C_1' \subseteq C_1$ and $C_2' \subseteq C_2$. Therefore by rule PPAR, $\Gamma' \vdash P_1 \mid P_2 \sim Q_1 \mid Q_2 \to C_1' \cup_\times C_2'$, and since $C_1' \cup_\times C_2' \subseteq C_1 \cup_\times C_2 (= C)$, this proves the claim.

If the last rule is POR, then there exist Γ'', x, T_1, T_2, C_1 and C_2 such that $\Gamma = \Gamma'', x : T_1 \vee T_2$, $C = C_1 \cup C_2$, $\Gamma'', x : T_1 \vdash P \sim Q \to C_1$ and $\Gamma'', x : T_2 \vdash P \sim Q \to C_2$. By definition of branches, it is clear that $\mathrm{branches}(\Gamma) = \mathrm{branches}(\Gamma'', x : T_1 \vee T_2) = \mathrm{branches}(\Gamma'', x : T_1) \cup \mathrm{branches}(\Gamma'', x : T_2)$. Thus, since $\Gamma' \in \mathrm{branches}(\Gamma)$, we know that $\Gamma' \in \mathrm{branches}(\Gamma'', x : T_1)$ or $\Gamma' \in \mathrm{branches}(\Gamma'', x : T_2)$. We write the proof for the case where $\Gamma' \in \mathrm{branches}(\Gamma'', x : T_1)$, the other case is analogous. By applying the induction hypothesis to $\Gamma'', x : T_1 \vdash P \sim Q \to C_1$, there exists $C_1' \subseteq C_1$ such that $\Gamma' \vdash P \sim Q \to C_1'$. Since $C_1 \subseteq C$, this proves the claim.

Lemma 10 (Environments in the constraints). *For all Γ, C, for all processes P, Q, if*

$$\Gamma \vdash P \sim Q \to C$$

then for all $(c, \Gamma') \in C$,

$$\mathrm{dom}(\Gamma') \subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P) \cup \mathrm{bvars}(Q) \cup \mathrm{nnames}(P) \cup \mathrm{nnames}(Q) \cup (\mathrm{nkeys}(P) \cup \mathrm{nkeys}(Q))^2$$

(where $\mathrm{bvars}(P)$, $\mathrm{nnames}(P)$, $\mathrm{nkeys}(P)$ respectively denote the sets of bound variables, names, and key names in P).

Proof. We prove this lemma by induction on the typing derivation of $\Gamma \vdash P \sim Q \to C$.

If the last rule applied in this derivation is PZERO, we have $C = \{(\emptyset, \Gamma)\}$, and the claim clearly holds.

In the PPAR case, we have $P = P_1 \mid P_2$, $Q = Q_1 \mid Q_2$, and $C = C_1 \cup_\times C_2$ for some P_1, P_2, Q_1, Q_2, C_1, C_2 such that $\Gamma \vdash P_1 \sim Q_1 \to C_1$ and $\Gamma \vdash P_2 \sim Q_2 \to C_2$. Thus any element of C is of the form $(c_1 \cup c_2, \Gamma_1 \cup \Gamma_2)$ where $(c_1, \Gamma_1) \in C_1$, $(c_2, \Gamma_2) \in C_2$, and Γ_1, Γ_2 are compatible. By the induction hypothesis,

$$\begin{aligned}\mathrm{dom}(\Gamma_1) &\subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P_1) \cup \mathrm{bvars}(Q_1) \cup \mathrm{nnames}(P_1) \cup \mathrm{nnames}(Q_1) \cup (\mathrm{nkeys}(P_1) \cup \mathrm{nkeys}(Q_1))^2 \\ &\subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P) \cup \mathrm{bvars}(Q) \cup \mathrm{nnames}(P) \cup \mathrm{nnames}(Q) \cup (\mathrm{nkeys}(P) \cup \mathrm{nkeys}(Q))^2,\end{aligned}$$

and similarly for Γ_2. Therefore, since $\mathrm{dom}(\Gamma_1 \cup \Gamma_2) = \mathrm{dom}(\Gamma_1) \cup \mathrm{dom}(\Gamma_2)$ (by definition), the claim holds.

In the PIN, PLET, PLETDEC, PLETADECSAME, and PLETADECDIFF cases, the typing judgement appearing in the condition of the rule uses Γ extended with an additional variable, which is bound in P and Q. We detail the PIN case, the other cases are similar. We have $P = \mathtt{in}(x).P'$, $Q = \mathtt{in}(x).Q'$ for some x, P', Q' such that $x \notin \mathrm{dom}(\Gamma)$ and $\Gamma, x : \mathtt{LL} \vdash P' \sim Q' \to C$. Hence by the induction hypothesis, if $(c, \Gamma') \in C$, then

$$\mathrm{dom}(\Gamma') \subseteq \mathrm{dom}(\Gamma, x : \mathtt{LL}) \cup \mathrm{bvars}(P') \cup \mathrm{bvars}(Q') \cup \mathrm{nnames}(P') \cup \mathrm{nnames}(Q') \cup (\mathrm{nkeys}(P') \cup \mathrm{nkeys}(Q'))^2.$$

Since $\mathrm{bvars}(P) = \{x\} \cup \mathrm{bvars}(P')$ and $\mathrm{bvars}(Q) = \{x\} \cup \mathrm{bvars}(Q')$, this proves the claim.

The cases of rules PNEW and PNEWKEY are similar, extending Γ with a nonce or key instead of a variable.

In the POUT case, there exist P', Q', M, N, C', c such that $P = \mathtt{out}(M).P'$, $Q = \mathtt{out}(N).Q'$, $C = C' \cup_\forall c$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma \vdash P' \sim Q' \to C'$. If $(c', \Gamma') \in C$, by definition of $\cup_\forall$ there exists c'' such that $(c'', \Gamma') \in C'$ and $c' = c \cup c''$. By the induction hypothesis, we thus have

$$\mathrm{dom}(\Gamma') \subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P') \cup \mathrm{bvars}(Q') \cup \mathrm{nnames}(P') \cup \mathrm{nnames}(Q')$$

and since $\mathrm{bvars}(P') = \mathrm{bvars}(P)$, $\mathrm{nnames}(P') = \mathrm{nnames}(P)$, and similarly for Q, this proves the claim.

In the PIFL case, there exist P', P'', Q', Q'', M, N, M', N', C', C'', c, c' such that $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$, $C = (C' \cup C'') \cup_\forall (c \cup c')$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$, $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'$, $\Gamma \vdash P' \sim Q' \to C'$, and $\Gamma \vdash P'' \sim Q'' \to C''$. If $(c'', \Gamma') \in C$, by definition of $\cup_\forall$ there exist c''', such that $(c''', \Gamma') \in C' \cup C''$ and $c'' = c''' \cup c \cup c'$. We write the proof for the case where $(c''', \Gamma') \in C'$, the other case is analogous. By the induction hypothesis, we thus have

$$\mathrm{dom}(\Gamma') \subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P') \cup \mathrm{bvars}(Q') \cup \mathrm{nnames}(P') \cup \mathrm{nnames}(Q') \cup (\mathrm{nkeys}(P') \cup \mathrm{nkeys}(Q'))^2$$

and since $\mathrm{bvars}(P') \subseteq \mathrm{bvars}(P)$, $\mathrm{nnames}(P') \subseteq \mathrm{nnames}(P)$, and similarly for Q, this proves the claim.

The cases of rules POR, PLETLRK, PIFLR, PIFS, PIFLR*, PIFP, PIFI, PIFLR'*, and PIFALL remain. All these cases are similar, we write the proof for the PIFLR* case. In this case, there exist P', P'', Q', Q'', M, N, M', N', C', C'', l, l', m, n such that

- $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$,
- $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$,
- $C = C' \cup C''$,
- $\Gamma \vdash M \sim N : [\![\tau_m^{l,\infty}\,;\, \tau_n^{l',\infty}]\!] \to \emptyset$,
- $\Gamma \vdash M' \sim N' : [\![\tau_m^{l,\infty}\,;\, \tau_n^{l',\infty}]\!] \to \emptyset$,
- $\Gamma \vdash P' \sim Q' \to C'$,
- and $\Gamma \vdash P'' \sim Q'' \to C''$.

If $(c, \Gamma') \in C$, we thus know that $(c, \Gamma') \in C'$ or $(c, \Gamma') \in C''$. We write the proof for the case where $(c, \Gamma') \in C'$, the other case is analogous. By the induction hypothesis, we thus have

$$\mathrm{dom}(\Gamma') \subseteq \mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P') \cup \mathrm{bvars}(Q') \cup \mathrm{nnames}(P') \cup \mathrm{nnames}(Q') \cup (\mathrm{nkeys}(P') \cup \mathrm{nkeys}(Q'))^2$$

and since $\mathrm{bvars}(P') \subseteq \mathrm{bvars}(P)$, $\mathrm{nnames}(P') \subseteq \mathrm{nnames}(P)$, and similarly for Q, this proves the claim.

Lemma 11 (Environments in the constraints do not contain union types). *For all Γ, C, for all processes P, Q, if*

$$\Gamma \vdash P \sim Q \to C$$

then for all $(c, \Gamma') \in C$,

$$\mathrm{branches}(\Gamma') = \{\Gamma'\}$$

i.e. *for all $x \in \mathrm{dom}(\Gamma')$, $\Gamma'(x)$ is not a union type.*

Proof. This property is immediate by induction on the typing derivation.

Lemma 12 (Typing is preserved by extending the environment). *For all Γ, Γ', P, Q, C, c, t, t', T, c, if $\Gamma \vdash \diamond$ and $\Gamma \cup \Gamma' \vdash \diamond$ (we do not require that Γ' is well-formed):*

- *if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \emptyset$, and if $\Gamma \vdash t \sim t' : T \to c$, then $\Gamma \cup \Gamma' \vdash t \sim t' : T \to c$.*
- *if $\mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') = \emptyset$, and if $\Gamma \vdash_d d(y) : T$, then $\Gamma \cup \Gamma' \vdash_d d(y) : T$.*

– *if* $(\mathrm{dom}(\Gamma) \cup \mathrm{bvars}(P) \cup \mathrm{bvars}(Q) \cup \mathrm{nnames}(P) \cup \mathrm{nnames}(Q)) \cap \mathrm{dom}(\Gamma') = \emptyset$, *and* $\mathrm{keys}(\Gamma') \cap (\mathrm{keys}(\Gamma) \cup \mathrm{nkeys}(P) \cup \mathrm{nkeys}(Q)) = \emptyset$, *and* $\Gamma \vdash P \sim Q \to C$, *then* $\Gamma \cup \Gamma' \vdash P \sim Q \to C'$. *where* $C' = \{(c, \Gamma_c \cup \Gamma'') | (c, \Gamma_c) \in C \wedge \Gamma'' \in \mathrm{branches}(\Gamma')\}$ *(note that the union is well defined,* i.e. Γ_c *and* Γ'' *are compatible, thanks to Lemma 10)*

Proof. – The first point is immediate by induction on the type derivation.
– The second point is immediate by examining the typing rules for destructors.
– The third point is immediate by induction on the type derivation of the processes. In the PZERO case, to satisfy the condition that the environment is its own only branch, rule POR needs to be applied first, in order to split all the union types in Γ', which yields the environments $\mathrm{branches}(\Gamma \cup \Gamma')$ in the constraints.

Lemma 13 (Consistency for Subsets). *The following statements about constraints hold:*

1. *If* (c, Γ) *is consistent, and* $c' \subseteq c$ *then* (c', Γ) *is consistent.*
2. *Let* C *be a consistent constraint set. Then every subset* $C' \subseteq C$ *is also consistent.*
3. *If* $C \cup_\forall c'$ *is consistent then* C *also is.*
4. *If* $C_1 \subseteq C_2$ *and* $C_1' \subseteq C_2'$, *then* $C_1 \cup_\times C_1' \subseteq C_2 \cup_\times C_2'$.
5. $[\![\cdot]\!]_{\sigma,\sigma'}$ *commutes with* $\cup$, $\cup_\times$, $\cup_\forall$, i.e. *for all* C, C', σ, σ', $[\![C \cup_\times C']\!]_{\sigma,\sigma'} = [\![C]\!]_{\sigma,\sigma'} \cup_\times [\![C']\!]_{\sigma,\sigma'}$ *and similarly for* $\cup$, $\cup_\forall$.
6. *If* σ_1 *and* σ_1' *are ground and have disjoint domains, as well as* σ_2 *and* σ_2', *then for all* c, $\left[\!\left[[\![c]\!]_{\sigma_1,\sigma_2} \right]\!\right]_{\sigma_1',\sigma_2'} = [\![c]\!]_{\sigma_1 \cup \sigma_1', \sigma_2 \cup \sigma_2'}$
7. *if* σ, σ' *are ground and* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma_\mathcal{X} \to c$ *for some* c, *if* $C \cup_\forall c$ *is consistent, and if for all* $(c', \Gamma') \in C$, $\Gamma \subseteq \Gamma'$, *then* $[\![C]\!]_{\sigma,\sigma'}$ *is consistent.*

Proof. Points 1 and 2 follow immediately from the definition of consistency and of static equivalence.

Point 3 follows from point 1: for every $(c, \Gamma) \in C$, $(c \cup c', \Gamma)$ is in $C \cup_\forall c'$, and is therefore consistent. Hence (c, Γ) also is by point 1.

Point 4 follows from the definition of $\cup_\times$. If $(c, \Gamma) \in C_1 \cup_\times C_1'$, there exists $(c_1, \Gamma_1) \in C_1$, $(c_1', \Gamma_1') \in C_1'$ such that $(c, \Gamma) = (c_1 \cup c_1', \Gamma_1 \cup \Gamma_1')$ (and Γ_1, Γ_1' are compatible). Since $C_1 \subseteq C_2$, $(c_1, \Gamma_1) \in C_2$. Similarly, $(c_1', \Gamma_1') \in C_2'$. Therefore $(c, \Gamma) \in C_2 \cup_\times C_2'$.

Points 5 and 6 follow from the definitions of $[\![\cdot]\!]_{\sigma,\sigma'}$, $\cup_\times$, $\cup_\forall$.

Point 7 follows from the definitions of $[\![\cdot]\!]_{\sigma,\sigma'}$, and of consistency. Indeed, let $(c', \Gamma') \in [\![C]\!]_{\sigma,\sigma'}$. There exists c'' such that $c' = [\![c'']\!]_{\sigma,\sigma'}$, and $(c'', \Gamma') \in C$. Let $c_1 \subseteq c'$, and $\Gamma_1 \subseteq \Gamma'$ such that ${\Gamma_1}_{\mathcal{N},\mathcal{K}} = {\Gamma'}_{\mathcal{N},\mathcal{K}}$ and $\mathrm{vars}(c_1) \subseteq \mathrm{dom}(\Gamma_1)$. Let θ, θ', c_θ be such that $(\Gamma_1)_{\mathcal{N},\mathcal{K}} \vdash \theta \sim \theta' : (\Gamma_1)_\mathcal{X} \to c_\theta$, and $c_\theta \subseteq [\![c_1]\!]_{\theta,\theta'}$.

Note that since σ, σ' are ground, c is also ground.

Since $c' = [\![c'']\!]_{\sigma,\sigma'}$, there exists $c_2 \subseteq c''$ such that $c_1 = [\![c_2]\!]_{\sigma,\sigma'}$. If we show that there exists c_3 such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma\theta \sim \sigma'\theta' : \Gamma_\mathcal{X} \to c_3$ and $c_3 \subseteq [\![c_2 \cup c]\!]_{\sigma\theta,\sigma'\theta'}$, then it will follow from the consistency of $C \cup_\forall c$ that $\phi_{\mathrm{LL}}^{\Gamma_2} \cup \phi_\ell(c_2 \cup c)\sigma\theta$ and $\phi_{\mathrm{LL}}^{\Gamma_2} \cup \phi_r(c_2 \cup c)\sigma'\theta'$ are statically equivalent, where $\Gamma_2 = \Gamma_1 \cup \Gamma \subseteq \Gamma''$.

Since ${\Gamma_1}_{\mathcal{N},\mathcal{K}} = {\Gamma''}_{\mathcal{N},\mathcal{K}}$ and $\Gamma \subseteq \Gamma''$, we have $\phi_{\mathrm{LL}}^{\Gamma_2} = \phi_{\mathrm{LL}}^{\Gamma_1}$. Hence $\phi_{\mathrm{LL}}^{\Gamma_1} \cup \phi_\ell(c_1 \cup c)\theta$ and $\phi_{\mathrm{LL}}^{\Gamma_1} \cup \phi_r(c_1 \cup c)\theta'$ will be statically equivalent.

Therefore, $\phi_{\mathrm{LL}}^{\Gamma_1} \cup \phi_\ell(c_1)\theta$ and $\phi_{\mathrm{LL}}^{\Gamma_1} \cup \phi_r(c_1)\theta'$ will also be statically equivalent, which proves the consistency of $[\![C]\!]_{\sigma,\sigma'}$.

It only remains to be proved that there exists c_3 such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma\theta \sim \sigma'\theta' : \Gamma_\mathcal{X} \to c_3$ and $c_3 \subseteq [\![c_2 \cup c]\!]_{\sigma\theta,\sigma'\theta'}$.

Since σ is ground, $\sigma\theta = \sigma \cup \theta|_{\mathrm{dom}(\Gamma_1) \setminus \mathrm{dom}(\Gamma)}$, and similarly for $\sigma'\theta'$. By assumption, $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma_\mathcal{X} \to c$ and $(\Gamma_1)_{\mathcal{N},\mathcal{K}} \vdash \theta \sim \theta' : (\Gamma_1)_\mathcal{X} \to c_\theta$. Hence there exists $c_3 \subseteq c \cup c_\theta$ such that $(\Gamma_2)_{\mathcal{N},\mathcal{K}} \vdash \sigma\theta \sim \sigma'\theta' : (\Gamma_2)_\mathcal{X} \to c_3$.

Since $c_\theta \subseteq [\![c_1]\!]_{\theta,\theta'}$, and c is ground, we have

$$\begin{aligned} c_3 &\subseteq c \cup [\![c_1]\!]_{\theta,\theta'} \\ &= c \cup [\![c_2]\!]_{\sigma\theta,\sigma'\theta'} \\ &= [\![c_2 \cup c]\!]_{\sigma\theta,\sigma'\theta'} \end{aligned}$$

which concludes the proof.

Lemma 14 (Environments in constraints contain a branch of the typing environment). *For all Γ, C, for all processes P, Q, if $\Gamma \vdash P \sim Q \to C$ then for all $(c, \Gamma') \in C$, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that $\Gamma'' \subseteq \Gamma'$.*

Proof. We prove this property by induction on the type derivation of $\Gamma \vdash P \sim Q \to C$. In the PZERO case, $C = \{(\emptyset, \Gamma)\}$, and by assumption $\mathrm{branches}(\Gamma) = \{\Gamma\}$, hence the claim trivially holds.

In the PPAR case, we have $P = P_1 \mid P_2$, $Q = Q_1 \mid Q_2$, and $C = C_1 \cup_\times C_2$ for some P_1, P_2, Q_1, Q_2, C_1, C_2 such that $\Gamma \vdash P_1 \sim Q_1 \to C_1$ and $\Gamma \vdash P_2 \sim Q_2 \to C_2$. Thus any element of C is of the form $(c_1 \cup c_2, \Gamma_1 \cup \Gamma_2)$ where $(c_1, \Gamma_1) \in C_1$, $(c_2, \Gamma_2) \in C_2$, and Γ_1, Γ_2 are compatible. By the induction hypothesis, both C_1 and C_2 contain a branch of Γ. The claim holds, as these are necessarily the same branch, since Γ_1 and Γ_2 are compatible.

In the POR case, we have $\Gamma = \Gamma'', x : T_1 \vee T_2$ for some x, Γ'', T_1, T_2 such that $\Gamma'', x : T_1 \vdash P \sim Q \to C_1$ and $\Gamma'', x : T_2 \vdash P \sim Q \to C_2$, and $C = C_1 \cup C_2$. Thus by the induction hypothesis, if $(c, \Gamma') \in C_i$ (for $i \in \{1, 2\}$), then Γ' contains some $\Gamma''' \in \mathrm{branches}(\Gamma'', x : T_i) \subseteq \mathrm{branches}(\Gamma)$, and the claim holds.

In the POUT case, there exist P', Q', M, N, C', c such that $P = \mathtt{out}(M).P'$, $Q = \mathtt{out}(N).Q'$, $C = C' \cup_\forall c$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma \vdash P' \sim Q' \to C'$. If $(c', \Gamma') \in C$, by definition of $\cup_\forall$ there exists c'' such that $(c'', \Gamma') \in C'$ and $c' = c \cup c''$. Hence by applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that $\Gamma'' \subseteq \Gamma'$.

In the PIFL case, there exist P', P'', Q', Q'', M, N, M', N', C', C'', c, c' such that $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$, $C = (C' \cup C'') \cup_\forall (c \cup c')$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$, $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'$, $\Gamma \vdash P' \sim Q' \to C'$, and $\Gamma \vdash P'' \sim Q'' \to C''$. If $(c'', \Gamma') \in C$, by definition of $\cup_\forall$ there exist c''', such that $(c''', \Gamma') \in C' \cup C''$ and $c'' = c''' \cup c \cup c'$. We write the proof for the case where $(c''', \Gamma') \in C'$, the other case is analogous. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that $\Gamma'' \subseteq \Gamma'$, which proves the claim.

All remaining cases are similar. We write the proof for the PIFLR* case. In this case, there exist P', P'', Q', Q'', M, N, M', N', C', C'', l, l', m, n such that $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$, $C = C' \cup C''$, $\Gamma \vdash M \sim N : [\![\tau_m^{l,\infty} ; \tau_n^{l',\infty}]\!] \to \emptyset$, $\Gamma \vdash M' \sim N' : [\![\tau_m^{l,\infty} ; \tau_n^{l',\infty}]\!] \to \emptyset$, $\Gamma \vdash P' \sim Q' \to C'$, and $\Gamma \vdash P'' \sim Q'' \to C''$. If $(c, \Gamma') \in C$, we thus know that $(c, \Gamma') \in C'$ or $(c, \Gamma') \in C''$. We write the proof for the case where $(c, \Gamma') \in C'$, the other case is analogous. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that $\Gamma'' \subseteq \Gamma'$, which proves the claim.

Lemma 15 (All branches are represented in the constraints). *For all Γ, C, for all processes P, Q, if $\Gamma \vdash P \sim Q \to C$ then for all $\Gamma' \in \mathrm{branches}(\Gamma)$, there exists $(c, \Gamma'') \in C$, such that $\Gamma' \subseteq \Gamma''$.*

Proof. We prove this property by induction on the type derivation of $\Gamma \vdash P \sim Q \to C$. In the PZERO case, $C = \{(\emptyset, \Gamma)\}$, and by assumption $\mathrm{branches}(\Gamma) = \{\Gamma\}$, hence the claim trivially holds.

In the PPAR case, we have $P = P_1 \mid P_2$, $Q = Q_1 \mid Q_2$, and $C = C_1 \cup_\times C_2$ for some P_1, P_2, Q_1, Q_2, C_1, C_2 such that $\Gamma \vdash P_1 \sim Q_1 \to C_1$ and $\Gamma \vdash P_2 \sim Q_2 \to C_2$. By the induction hypothesis, there exists

$(c_1, \Gamma_1) \in C_1$ and $(c_2, \Gamma_2) \in C_2$ such that $\Gamma' \subseteq \Gamma_1$ and $\Gamma' \subseteq \Gamma_2$. By Lemma 10, $\mathrm{dom}(\Gamma_1)$ and $\mathrm{dom}(\Gamma_2)$ only contain $\mathrm{dom}(\Gamma)(= \mathrm{dom}(\Gamma'))$ and variables and names in

$$\mathrm{bvars}(P_1) \cup \mathrm{bvars}(Q_1) \cup \mathrm{nnames}(P_1) \cup \mathrm{nnames}(Q_1) \cup (\mathrm{nkeys}(P_1) \cup \mathrm{nkeys}(Q_1))^2$$

and

$$\mathrm{bvars}(P_2) \cup \mathrm{bvars}(Q_2) \cup \mathrm{nnames}(P_2) \cup \mathrm{nnames}(Q_2) \cup (\mathrm{nkeys}(P_2) \cup \mathrm{nkeys}(Q_2))^2$$

respectively. We have $\Gamma_1(x) = \Gamma_2(x) = \Gamma'(x)$ for all $x \in \mathrm{dom}(\Gamma')$, and the sets

$$\mathrm{bvars}(P_1) \cup \mathrm{bvars}(Q_1) \cup \mathrm{nnames}(P_1) \cup \mathrm{nnames}(Q_1) \cup (\mathrm{nkeys}(P_1) \cup \mathrm{nkeys}(Q_1))^2$$

and

$$\mathrm{bvars}(P_2) \cup \mathrm{bvars}(Q_2) \cup \mathrm{nnames}(P_2) \cup \mathrm{nnames}(Q_2) \cup (\mathrm{nkeys}(P_2) \cup \mathrm{nkeys}(Q_2))^2$$

are disjoint by well formedness of the processes $P_1 \mid P_2$ and $Q_1 \mid Q_2$. Thus Γ_1 and Γ_2 are compatible. Therefore $(c_1 \cup c_2, \Gamma_1 \cup \Gamma_2) \in C_1 \cup_\times C_2 (= C)$, and the claim holds since $\Gamma' \subseteq \Gamma_1 \cup \Gamma_2$.

In the POR case, we have $\Gamma = \Gamma'', x : T_1 \vee T_2$ for some x, Γ'', T_1, T_2 such that $\Gamma'', x : T_1 \vdash P \sim Q \to C_1$ and $\Gamma'', x : T_2 \vdash P \sim Q \to C_2$, and $C = C_1 \cup C_2$. Since $\mathrm{branches}(\Gamma) = \mathrm{branches}(\Gamma'', x : T_1) \cup \mathrm{branches}(\Gamma'', x : T_2)$, we know that $\Gamma' \in \mathrm{branches}(\Gamma'', x : T_i)$ for some i. We conclude this case directly by applying the induction hypothesis to $\Gamma'', x : T_i \vdash P \sim Q \to C_i$.

In the POUT case, there exist P', Q', M, N, C', c such that $P = \mathtt{out}(M).P'$, $Q = \mathtt{out}(N).Q'$, $C = C' \cup_\forall c$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma \vdash P' \sim Q' \to C'$. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $(c'', \Gamma'') \in C'$ such that $\Gamma' \subseteq \Gamma''$. By definition of $\cup_\forall$, $(c' \cup c, \Gamma'') \in C$, which proves the claim.

In the PIFL case, there exist $P', P'', Q', Q'', M, N, M', N', C', C'', c, c'$ such that $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$, $C = (C' \cup C'') \cup_\forall (c \cup c')$, $\Gamma \vdash M \sim N : \mathtt{LL} \to c$, $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'$, $\Gamma \vdash P' \sim Q' \to C'$, and $\Gamma \vdash P'' \sim Q'' \to C''$. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $(c'', \Gamma'') \in C'$ such that $\Gamma' \subseteq \Gamma''$. By definition of $\cup_\forall$, $(c'' \cup c \cup c', \Gamma'') \in C$, which proves the claim.

All remaining cases are similar. We write the proof for the PIFLR* case. In this case, there exist P', P'', $Q', Q'', M, N, M', N', C', C'', l, l', m, n$ such that $P = \mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = N'\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$, $C = C' \cup C''$, $\Gamma \vdash M \sim N : [\![\tau_m^{l,\infty}\,;\,\tau_n^{l',\infty}]\!] \to \emptyset$, $\Gamma \vdash M' \sim N' : [\![\tau_m^{l,\infty}\,;\,\tau_n^{l',\infty}]\!] \to \emptyset$, $\Gamma \vdash P' \sim Q' \to C'$, and $\Gamma \vdash P'' \sim Q'' \to C''$. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $(c'', \Gamma'') \in C'$ such that $\Gamma' \subseteq \Gamma''$.

If $(c, \Gamma') \in C$, we thus know that $(c, \Gamma') \in C'$ or $(c, \Gamma') \in C''$. We write the proof for the case where $(c, \Gamma') \in C'$, the other case is analogous. By applying the induction hypothesis to $\Gamma \vdash P' \sim Q' \to C'$, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that $\Gamma'' \subseteq \Gamma'$, which proves the claim.

Lemma 16 (Refinement types). *For all Γ, for all terms t, t', for all m, n, a, l, l', c, if $\Gamma \vdash t \sim t' : [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \to c$ then $c = \emptyset$ and*

- *either $t = m$, $t' = n$, $a = \infty$ and $\Gamma(m) = \tau_m^{l,a}$ and $\Gamma(n) = \tau_n^{l',a}$;*
- *or $t = m$, $t' = n$, $a = 1$, and $(\Gamma(m) = \tau_m^{l,a}) \vee (m \in \mathcal{FN} \cup \mathcal{C} \wedge l = \mathtt{LL})$, and $(\Gamma(n) = \tau_n^{l',a}) \vee (n \in \mathcal{FN} \cup \mathcal{C} \wedge l' = \mathtt{LL})$;*
- *or t and t' are variables $x, y \in \mathcal{X}$ and there exist labels l'', l''', and names m', n' such that $\Gamma(x) = [\![\tau_m^{l,a}\,;\,\tau_{n'}^{l'',a}]\!]$ and $\Gamma(y) = [\![\tau_{m'}^{l''',a}\,;\,\tau_n^{l',a}]\!]$.*

In particular if t, t' are ground then the last case cannot occur.

Proof. The proof of this property is immediate by induction on the typing derivation for the terms. Indeed, because of the form of the type, and by well-formedness of Γ, the only rules which can lead to $\Gamma \vdash t \sim t' : [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!] \to c$ are TVAR, TLR^1, TLR$^\infty$, TLRVAR, and TSUB.

In the TVAR, TLR^1, TLR$^\infty$ cases the claim directly follows from the premises of the rule.

In the TLRVAR case, t and t' are necessarily variables, and their types in Γ are obtained directly by applying the induction hypothesis to the premises of the rule.

Finally in the TSUB case, $\Gamma \vdash t \sim t' : T \to c$ and $T <: [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!]$. By Lemma 3, $T = [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!]$ and we conclude by the induction hypothesis.

Lemma 17 (Encryption types). *For all environment Γ, types T, T', messages M, N, M_1, M_2 and set of constraints c:*

1. *If $\Gamma \vdash M \sim N : (T)_{T'} \to c$ then*
   - *either there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $M = \mathtt{enc}(M_1, M_2)$, $N = \mathtt{enc}(N_1, N_2)$, $c = c_1 \cup c_2$, $\Gamma \vdash M_1 \sim N_1 : T \to c_1$, and $\Gamma \vdash M_2 \sim N_2 : T' \to c_2$, both with shorter derivations (than the one for $\Gamma \vdash M \sim N : (T)_{T'} \to c$);*
   - *or M and N are variables.*
2. *If $\Gamma \vdash M \sim N : \{T\}_{T'} \to c$ then*
   - *either there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $M = \mathtt{aenc}(M_1, M_2)$, $N = \mathtt{aenc}(N_1, N_2)$, $c = c_1 \cup c_2$, $\Gamma \vdash M_1 \sim N_1 : T \to c_1$ and $\Gamma \vdash M_2 \sim N_2 : T' \to c_2$, both with shorter derivations (than the one for $\Gamma \vdash M \sim N : \{T\}_{T'} \to c$);*
   - *or M and N are variables.*
3. *If $T <: \mathtt{LL}$ and $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : T \to c$ then $T = \mathtt{LL}$.*
4. *If $T <: \mathtt{LL}$ and $\Gamma \vdash \mathtt{aenc}(M_1, M_2) \sim N : T \to c$ then $T = \mathtt{LL}$.*
5. *If $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : \mathtt{LL} \to c$ then there exist N_1, N_2 such that $N = \mathtt{enc}(N_1, N_2)$, and*
   - *either there exist T', c_1, c_2 such that $\Gamma \vdash M_2 \sim N_2 : \mathrm{key}^{\mathtt{HH}}(T') \to c_2$, $c = \{\mathtt{enc}(M_1, M_2) \sim N\} \cup c_1 \cup c_2$, and $\Gamma \vdash M_1 \sim N_1 : T' \to c_1$;*
   - *or there exist c_1, c_2 such that $c = c_1 \cup c_2$, $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$ and $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$.*
6. *If $\Gamma \vdash \mathtt{aenc}(M_1, M_2) \sim N : \mathtt{LL} \to c$ then there exist N_1, N_2 such that $N = \mathtt{aenc}(N_1, N_2)$, and*
   - *either there exist T', T'', c_1, c_2 such that $T' <: \mathrm{key}^{\mathtt{HH}}(T'')$, $\Gamma \vdash M_2 \sim N_2 : \mathrm{pkey}(T') \to c_2$, $c = \{\mathtt{aenc}(M_1, M_2) \sim N\} \cup c_1 \cup c_2$, and $\Gamma \vdash M_1 \sim N_1 : T'' \to c_1$;*
   - *or there exist c_1, c_2 such that $c = c_1 \cup c_2$, $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$, and $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$.*
7. *The symmetric properties to the previous four points,* i.e. *when the term on the right is an encryption, also hold.*

Proof. We prove point 1 by induction on the derivation of $\Gamma \vdash M \sim N : (T)_{T'} \to c$. Because of the form of the type, and by well-formedness of Γ, the only possibilities for the last rule applied are TVAR, TENC, and TSUB. The claim clearly holds in the TVAR and TENC cases. In the TSUB case, we have $\Gamma \vdash M \sim N : T'' \to c$ for some $T'' <: (T)_{T'}$. Hence by Lemma 3, there exists $T''' <: T$ such that $T'' = (T''')_{T'}$. Therefore, by applying the induction hypothesis to $\Gamma \vdash M \sim N : T'' \to c$

- either M and N are two variables, and the claim holds;
- or there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $M = \mathtt{enc}(M_1, M_2)$, $N = \mathtt{enc}(N_1, N_2)$, $c = c_1 \cup c_2$, $\Gamma \vdash M_1 \sim N_1 : T''' \to c_1$, and $\Gamma \vdash M_2 \sim N_2 : T' \to c_2$, both with shorter derivations than the one for $\Gamma \vdash M \sim N : T'' \to c$. Thus by subtyping (rule TSUB), $\Gamma \vdash M_1 \sim N_1 : T \to c_1$ with a shorter derivation that $\Gamma \vdash M \sim N : (T)_{T'} \to c$, which proves the property.

Point 2 has a similar proof to point 1.

We now prove point 3 by induction on the proof of $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : T \to c$. Because of the form of the terms, the last rule applied can only be THIGH, TOR, TENC, TENCH, TENCL, TAENCH, TAENCL, TLR', TLRL' or TSUB.

The THIGH, TLR', TOR, TENC cases are actually impossible by Lemma 3, since $T <: \mathtt{LL}$. In the TSUB case, we have $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : T' \to c$ for some T' such that $T' <: T$. By transitivity of $<:$, $T' <: \mathtt{LL}$, and the induction hypothesis proves the claim. In all other cases, $T = \mathtt{LL}$ and the claim holds.

Point 4 has a similar proof to point 3.

We prove point 5 by induction on the proof of $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : \mathtt{LL} \to c$. Because of the form of the terms and of the type (*i.e.* LL) the last rule applied can only be TENCH, TENCL, TAENCH, TAENCL, TLRL' or TSUB.

The TLRL' case is impossible, since by Lemma 16 it would imply that $\mathtt{enc}(M_1, M_2)$ is either a variable or a nonce.

In the TSUB case, we have $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : T' \to c$ for some T' such that $T' <: \mathtt{LL}$. By point 3, $T' = \mathtt{LL}$, and the premise of the rule thus gives a shorter derivation of $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : \mathtt{LL} \to c$. The induction hypothesis applied to this shorter derivation proves the claim.

The TAENCH and TAENCL cases are impossible, since the condition of the rule would then imply $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : \{T\}_{T'} \to c'$ for some T, T', c', which is not possible by point 2.

In the TENCH case, there exist T, T', c' such that $c = \{\mathtt{enc}(M_1, M_2) \sim N\} \cup c'$, $T' <: \mathrm{key}^{\mathtt{HH}}(T)$, and $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : (T)_{T'} \to c'$. By point 1, since $\mathtt{enc}(M_1, M_2)$ is not a variable, there exist N_1, N_2, c_1, c_2 such that $N = \mathtt{enc}(N_1, N_2)$, $c' = c_1 \cup c_2$, $\Gamma \vdash M_1 \sim N_1 : T \to c_1$, and $\Gamma \vdash M_2 \sim N_2 : T' \to c_2$. Hence by subtyping, $\Gamma \vdash M_2 \sim N_2 : \mathrm{key}^{\mathtt{HH}}(T) \to c_2$.

Finally in the TENCL case, there similarly exist T, T' such that $T <: \mathrm{key}^{\mathtt{LL}}(T')$ or $T = \mathtt{LL}$, and $\Gamma \vdash \mathtt{enc}(M_1, M_2) \sim N : (\mathtt{LL})_T \to c$. Hence by point 1, there exist N_1, N_2, c_1, c_2 such that $N = \mathtt{enc}(N_1, N_2)$, $c = c_1 \cup c_2$ $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$, and $\Gamma \vdash M_2 \sim N_2 : T \to c_2$. Hence in any case (potentially using subtyping), $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$, which proves the claim.

Point 6 has a similar proof to point 5.

The symmetric properties, as described in point 7, have analogous proofs.

Lemma 18 (Signature types). *For all environment Γ, type T, messages M_1, M_2, N, and set of constraints c:*

1. *If $T <: \mathtt{LL}$ and $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : T \to c$ then $T = \mathtt{LL}$.*
2. *If $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : \mathtt{LL} \to c$ then there exist N_1, N_2 such that $N = \mathtt{sign}(N_1, N_2)$, and*
   - *either there exist T, c' and c'' such that $\Gamma \vdash M_2 \sim N_2 : \mathrm{eqkey}^{\mathtt{HH}}(T) \to \emptyset$, $c = \{\mathtt{sign}(M_1, M_2) \sim N\} \cup c' \cup c''$, $\Gamma \vdash M_1 \sim N_1 : T \to c'$, and $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c''$;*
   - *or there exists c_1, c_2 such that $c = c_1 \cup c_2$, $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$ and $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$.*
3. *The symmetric properties to the previous points,* i.e. *when the term on the right is a signature, also hold.*

Proof. We prove point 1 by induction on the proof of $\Gamma \vdash \mathtt{sign}(M, k) \sim N : T \to c$. Because of the form of the terms, the last rule applied can only be THIGH, TOR, TENCH, TENCL, TAENCH, TAENCL, TSIGNH, TSIGNL, TLR', TLRL' or TSUB.

The THIGH, TLR', TOR cases are actually impossible by Lemma 3, since $T <: \mathtt{LL}$. In the TSUB case, we have $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : T' \to c$ for some T' such that $T' <: T$. By transitivity of $<:$, $T' <: \mathtt{LL}$, and the induction hypothesis proves the claim. In all other cases, $T = \mathtt{LL}$ and the claim holds.

We prove point 2 by induction on the proof of $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : \mathtt{LL} \to c$. Because of the form of the terms and of the type (*i.e.* LL) the last rule applied can only be TENCH, TENCL, TAENCH, TAENCL, TSIGNH, TSIGNL, TLRL' or TSUB.

The TLRL' case is impossible, since by Lemma 16 it would imply that $\mathtt{sign}(M_1, M_2)$ is a variable or a nonce.

In the TSUB case, we have $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : T \to c$ for some T such that $T <: \mathtt{LL}$. By point 1, $T = \mathtt{LL}$, and the premise of the rule thus gives a shorter derivation of $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : \mathtt{LL} \to c$. The induction hypothesis applied to this shorter derivation proves the claim.

The TENCH, TENCL, TAENCH and TAENCL cases are impossible, since the condition of the rule would then imply $\Gamma \vdash \mathtt{sign}(M_1, M_2) \sim N : (T)_{T'} \to c'$ (or $\{T\}_{T'}$) for some T, T', c', which is not possible by Lemma 17.

Finally, in the TSIGNH and TSIGNL cases, the premises of the rule directly proves the claim.

The symmetric properties, as described in point 3, have analogous proofs.

Lemma 19 (Pair types). *For all environment Γ, for all M, N, T, c:*

1. *For all T_1, T_2, if $\Gamma \vdash M \sim N : T_1 * T_2 \to c$ then*
   - *either there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $M = \langle M_1, M_2 \rangle$, $N = \langle N_1, N_2 \rangle$, $c = c_1 \cup c_2$, and $\Gamma \vdash M_1 \sim N_1 : T_1 \to c_1$ and $\Gamma \vdash M_2 \sim N_2 : T_2 \to c_2$;*
   - *or M and N are variables.*
2. *For all M_1, M_2, if $T <: \mathtt{LL}$ and $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : T \to c$ then either $T = \mathtt{LL}$ or there exists T_1, T_2 such that $T = T_1 * T_2$.*
3. *For all M_1, M_2, if $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : \mathtt{LL} \to c$ then there exist N_1, N_2, c_1, c_2, such that $c = c_1 \cup c_2$, $N = \langle N_1, N_2 \rangle$, $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$ and $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$.*
4. *The symmetric properties to the previous two points (*i.e. *when the term on the right is a pair) also hold.*

Proof. Let us prove point 1 by induction on the typing derivation $\Gamma \vdash M \sim N : T_1 * T_2 \to c$. Because of the form of the type, and by well-formedness of Γ, the only possibilities for the last rule applied are TVAR, TPAIR, and TSUB.

The claim clearly holds in the TVAR and TPAIR cases.

In the TSUB case, $\Gamma \vdash M \sim N : T' \to c$ for some $T' <: T_1 * T_2$, and by Lemma 3, $T' = T'_1 * T'_2$ for some T'_1, T'_2 such that $T'_1 <: T_1$ and $T'_2 <: T_2$. Therefore, by applying the induction hypothesis to $\Gamma \vdash M \sim N : T'_1 * T'_2 \to c$, M and N are either two variables, and the claim holds; or two pairs, *i.e.* there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $M = \langle M_1, M_2 \rangle$, $N = \langle N_1, N_2 \rangle$, $c = c_1 \cup c_2$, and for $i \in \{1, 2\}$, $\Gamma \vdash M_i \sim N_i : T'_i \to c_i$. Hence, by subtyping, $\Gamma \vdash M_i \sim N_i : T_i \to c_i$, and the claim holds.

We now prove point 2 by induction on the proof of $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : T \to c$. Because of the form of the terms, the last rule applied can only be THIGH, TOR, TPAIR, TENCH, TENCL, TAENCH, TAENCL, TLR', TLRL' or TSUB.

The THIGH, TLR', and TOR cases are actually impossible by Lemma 3, since $T <: \mathtt{LL}$.

The TLRL' and case is also impossible, since by Lemma 16 it would imply that $\langle M_1, M_2 \rangle$ is either a variable or a nonce.

The TENCH, TENCL, TAENCH, TAENCL cases are impossible, since the condition of the rule would then imply $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : (T)_{T'} \to c'$ (or $\{T\}_{T'}$) for some T, T', c', which is not possible by Lemma 17.

In the TPAIR case, the claim clearly holds.

Finally, in the TSUB case, we have $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : T' \to c$ for some T' such that $T' <: T$. By transitivity of $<:$, $T' <: \mathtt{LL}$, and we may apply the induction hypothesis to $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : T' \to c$. Hence

either $T' = \mathtt{LL}$ or $T' = T'_1 * T'_2$ for some T'_1, T'_2. By Lemma 3, this implies in the first case that $T = \mathtt{LL}$ and in the second case that $T = \mathtt{LL}$ or T is also a pair type ($T \neq \mathtt{HL}$ and $T \neq \mathtt{HH}$ in both cases, since we already know that $T <: \mathtt{LL}$).

We prove point 3 as a consequence of the first two points, by induction on the derivation of $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : \mathtt{LL} \to c$. The last rule in this derivation can only be TENCH, TENCL, TAENCH, TAENCL, TLR', TLRL' or TSUB by the form of the types and terms, but similarly to the previous point TENCH, TENCL, TAENCH, TAENCL, TLR' and TLRL' are actually not possible.

Hence the last rule of the derivation is TSUB. We have $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : T \to c$ for some T such that $T <: \mathtt{LL}$. By point 2, either $T = \mathtt{LL}$ or there exist T_1, T_2 such that $T = T_1 * T_2$. If $T = \mathtt{LL}$, we have a shorter proof of $\Gamma \vdash \langle M_1, M_2 \rangle \sim N : \mathtt{LL} \to c$ and we conclude by the induction hypothesis. Otherwise, since $T <: \mathtt{LL}$, by Lemma 3, $T_1 <: \mathtt{LL}$ and $T_2 <: \mathtt{LL}$. Moreover by the first property, there exist N_1, N_2, c_1, c_2 such that $N = \langle N_1, N_2 \rangle$, $c = c_1 \cup c_2$, $\Gamma \vdash M_1 \sim N_1 : T_1 \to c_1$, and $\Gamma \vdash M_2 \sim N_2 : T_2 \to c_2$.

Thus by subtyping, $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1$ and $\Gamma \vdash M_2 \sim N_2 : \mathtt{LL} \to c_2$, which proves the claim.

The symmetric properties, as described in point 4, have analogous proofs.

Lemma 20 (Type for keys, nonces and constants). *For all well-formed environment Γ, for all messages M, N, for all key $k \in \mathcal{K}$, for all nonce or constant $n \in \mathcal{N} \cup \mathcal{C}$, for all c, l, the following properties hold:*

1. *For all T, T', if $\Gamma \vdash M \sim N : T \to c$ and $T <: \mathrm{key}^l(T')$, then $c = \emptyset$; and either M, N are in $\mathcal{BK}$ and $\Gamma(M, N) <: T$; or M and N are variables.*
2. *If $\Gamma \vdash M \sim N : \mathrm{pkey}(T) \to c$, then $c = \emptyset$; and either $\exists M', N'.M = \mathtt{pk}(M') \wedge N = \mathtt{pk}(N') \wedge \Gamma \vdash M' \sim N' : T \to \emptyset$; or M, N are variables.*
3. *If $\Gamma \vdash M \sim N : \mathrm{vkey}(T) \to c$, then $c = \emptyset$; and either $\exists M', N'.M = \mathtt{vk}(M') \wedge N = \mathtt{vk}(N') \wedge \Gamma \vdash M' \sim N' : T \to \emptyset$; or M, N are variables.*
4. *If $l \in \{\mathtt{LL}, \mathtt{HH}\}$, and $\Gamma \vdash k \sim N : l \to c$, then $N \in \mathcal{K}$, $c = \emptyset$, and either $k, N \in \mathcal{BK}$ and there exists T such that $\Gamma(k, N) <: \mathrm{key}^l(T)$, or $l = \mathtt{LL}$, and $k = N \in \mathcal{FK}$.*
5. *If $\Gamma \vdash \mathtt{pk}(M') \sim N : \mathtt{LL} \to c$, then $c = \emptyset$, $\exists N'.N = \mathtt{pk}(N')$, and either $\exists l, T'.T <: \mathrm{eqkey}^l(T') \wedge \Gamma \vdash M' \sim N' : T \to \emptyset$, or $\exists k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}.M' = N' = k$.*
6. *If $\Gamma \vdash \mathtt{vk}(M') \sim N : \mathtt{LL} \to c$, then $c = \emptyset$, $\exists N'.N = \mathtt{vk}(N')$, and either $\exists l, T'.T <: \mathrm{eqkey}^l(T') \wedge \Gamma \vdash M' \sim N' : T \to \emptyset$, or $\exists k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}.M' = N' = k$.*
7. *If $\Gamma \vdash n \sim N : \mathtt{HH} \to c$, then $n \in \mathcal{BN}$, $c = \emptyset$ and either $\Gamma(n) = \tau_n^{\mathtt{HH},1}$ or $\tau_n^{\mathtt{HH},\infty}$.*
8. *If $\Gamma \vdash n \sim N : \mathtt{LL} \to c$, then $N = n$, $c = \emptyset$, and either there exists $a \in \{1, \infty\}$ such that $\Gamma(n) = \tau_n^{\mathtt{LL},a}$, or $n \in \mathcal{FN} \cup \mathcal{C}$.*
9. *The symmetric properties to points 4 to 8 (*i.e. *with k (resp. $\mathtt{pk}(M')$, $\mathtt{vk}(M')$, n) on the right) also hold.*

Proof. Point 1 is easily proved by induction on the derivation of $\Gamma \vdash M \sim N : T \to c$. Indeed, by the form of the type, using Lemma 3, the last rule can only be TKEY, TVAR, or TSUB. In the TKEY and TVAR cases the claim clearly holds. In the TSUB case, we have $T'' <: T$ such that $\Gamma \vdash M \sim N : T' \to c$ with a shorter derivation. By transitivity, $T'' <: \mathrm{key}^l(T')$. Thus by applying by the induction hypothesis, either M, N are variables, or they are keys and $\Gamma(M, N) <: T'' <: T$, and in both cases the claim holds.

Similarly, we prove point 2 by induction on the derivation of $\Gamma \vdash M \sim N : \mathrm{pkey}(T) \to c$. Indeed, by the form of the type, and since Γ is well-formed, the last rule can only be TPUBKEY, TVAR, or TSUB. In the TPUBKEY and TVAR cases the claim clearly holds. In the TSUB case, we have $T' <: \mathrm{pkey}(T)$ such that $\Gamma \vdash M \sim N : T' \to c$ with a shorter derivation. By Lemma 3, $T' = \mathrm{pkey}(T)$. We conclude the proof by applying by the induction hypothesis to the shorter derivation of $\Gamma \vdash M \sim N : T' \to c$.

Point 3 has a similar proof to point 2.

We prove point 4 by induction on the derivation of $\Gamma \vdash k \sim N : l \to c$. Because of the form of the terms and type, and by well-formedness of Γ, the last rule applied can only be TCstFN, TEncH, TEncL, TAencH, TAencL, TLR', TLRL' or TSub.

In the TCstFN case, $k = N \in \mathcal{FK}$ and the claim holds.

The TEncH, TEncL, TAencH, TAencL cases are impossible since they would imply that $\Gamma \vdash k \sim N : (T')_{k',k''} \to c'$ (or $\{T'\}_{k',k''}$) for some T', k', k'', c', which is impossible by Lemma 17.

The TLR' and TLRL' cases are impossible. Indeed in these cases, we have $\Gamma \vdash k \sim N : [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!] \to \emptyset$ for some m, n. Lemma 16 then implies that $m = k$ (and $n = N$), which is contradictory.

Finally, in the TSub case, we have $\Gamma \vdash k \sim N : T \to c$ for some T such that $T <: l$. By Lemma 3, this implies that T is either a pair type, an encryption type, a public or verification key type, a key type, or l. The pair, encryption, public and verification key cases are impossible, respectively by Lemma 19, Lemma 17, and point 2, since $k \in \mathcal{K}$. The last case is trivial by the induction hypothesis. Only the case where $T <: \mathrm{key}^l(T')$ (for some T') remains. By point 1, in that case, since k is not a variable, we have $N \in \mathcal{K}$ and $\Gamma(k, N) <: \mathrm{key}^l(T')$, and therefore the claim holds.

Similarly, we prove point 5 by induction on the derivation of $\Gamma \vdash \mathtt{pk}(M') \sim N : \mathtt{LL} \to c$. Because of the form of the terms and type, and by well-formedness of Γ, the last rule applied can only be TEncH, TEncL, TAencH, TAencL, TLRL', TPubKeyL or TSub.

The TEncH, TEncL, TAencH, TAencL cases are impossible since they would imply that $\Gamma \vdash \mathtt{pk}(M') \sim N : (T')_k \to c'$ (or $\{T'\}_{k'}$) for some T', k, c', which is impossible by Lemma 17.

The TLRL' case is also impossible. Indeed in this case, we have $\Gamma \vdash \mathtt{pk}(M') \sim N : [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!] \to \emptyset$ for some m, n. Lemma 16 then implies that $m = \mathtt{pk}(M')$ (and $n = N$), which is contradictory.

In the TSub case, we have $\Gamma \vdash \mathtt{pk}(M') \sim N : T \to c$ for some T such that $T <: \mathtt{LL}$. By Lemma 3, this implies that T is either a pair type, an encryption type, a public or verification key type, a key type, or $\mathtt{LL}$. Just as in the previous point, the pair, encryption, verification key, and key cases are impossible. If $T = \mathtt{LL}$, the claim trivially holds by the induction hypothesis. The case where $T = \mathrm{pkey}(T')$ (for some T') remains. Since $\Gamma \vdash \mathtt{pk}(M') \sim N : T \to c$, by point 2 we have $N = \mathtt{pk}(N')$ for some N', $c = \emptyset$ and $\Gamma \vdash M' \sim N' : T' \to \emptyset$. In addition, by Lemma 3, since $T <: \mathtt{LL}$, there exist l, T'' such that $T' <: \mathrm{eqkey}^l(T'')$, and the claim holds.

Finally in the TPubKeyL case, the claim clearly holds.

Point 6 has a similar proof to point 5.

The remaining properties have similar proofs to point 4. For point 7, *i.e.* if $\Gamma \vdash n \sim t : \mathtt{HH} \to c$, only the TNonce, TSub, and TLR' cases are possible. The claim clearly holds in the TNonce case.

In the TLR' case, we have $\Gamma \vdash n \sim t : [\![\tau_m^{\mathtt{HH},a} ; \tau_p^{\mathtt{HH},a}]\!] \to \emptyset$ for some m, p. Lemma 16 then implies that $m = n$, and $p = t$, and $\Gamma(n) = \tau_m^{\mathtt{HH},a}$, and $\Gamma(p) = \tau_p^{\mathtt{HH},a}$, which proves the claim.

In the TSub case, $\Gamma \vdash n \sim t : T \to c$ for some $T <: \mathtt{HH}$, thus by Lemma 3, T is either a pair type (impossible by Lemma 19), an encryption type (impossible by Lemma 17), a public key, verification key, or key type (impossible by points 1 to 3), or $\mathtt{HH}$ (and we conclude by the induction hypothesis).

For point 8, similarly, only the TNonceL, TCstFN, TSub, TLRL' cases are possible. The TSub case is proved in the same way as for the third property. The TLRL' case is proved similarly to the previous point. Finally the claim clearly holds in the TNonceL and TCstFN cases.

The symmetric properties, as described in point 9, have analogous proofs.

Lemma 21 (Application of destructors). *For all Γ, for all t, t', T, c, for all ground substitutions σ, σ' such that $\mathrm{dom}(\sigma) = \mathrm{dom}(\sigma') = \mathrm{vars}(t) \cup \mathrm{vars}(t')$, if $\Gamma \vdash_d t \sim t' : T$ and ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c$, where $\Gamma' = \Gamma_{\mathcal{N},\mathcal{K}} \cup \Gamma|_{\mathrm{dom}(\sigma)}$, then:*

1. *We have:*
$$(t\sigma)\downarrow = \bot \iff (t'\sigma')\downarrow = \bot$$
2. *And if $(t\sigma)\downarrow \neq \bot$ then there exists $c' \subseteq c$ such that*
$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash (t\sigma)\downarrow \sim (t'\sigma')\downarrow : T \to c'$$

Proof. Since $\Gamma \vdash_d t \sim t' : T$, by examining the typing rules for destructors, we can distinguish four cases for t, t'.

– $t = \mathtt{dec}(x, M)$ and $t' = \mathtt{dec}(x, M)$, for some variable $x \in \mathcal{X}$, and some $M \in \mathcal{X} \cup \mathcal{K}$. We know that $\Gamma \vdash_d t \sim t' : T$, which can be proved using the rule DDecL, DDecT, DDecT', DDecH', or DDecL'. In the DDecL, DDecH', DDecL' cases, $\Gamma(x) = \mathtt{LL}$, and in the DDecT and DDecT' cases $\Gamma(x) = (T)_{T'}$ for some T'. By assumption we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \Gamma(x) \to c_x$ for some $c_x \subseteq c$.

- Let us prove 1) by contraposition. Assume $t\sigma\downarrow \neq \bot$. Hence, $\sigma(x) = \mathtt{enc}(M', M\sigma)$ for some M', and $M\sigma$ is a key $k \in \mathcal{K}$. We will now show that $M\sigma' = M\sigma = k$. It is clear if $M \in \mathcal{K}$. Otherwise, $M \in \mathcal{X}$, and:
  * In the DDecL and DDecL' cases, $\Gamma \vdash M \sim M : \mathtt{LL} \to \emptyset$. Since σ, σ' are well-typed, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma \sim M\sigma' : \mathtt{LL} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash k \sim M\sigma' : \mathtt{LL} \to c''$. Thus by Lemma 20, either $M\sigma' = k \in \mathcal{FK}$; or $M\sigma' \in \mathcal{BK}$ and $\Gamma(k, M\sigma') <: \mathrm{key}^{\mathtt{LL}}(T''')$ for some T'''. Hence, in either case (using the well-formedness of Γ in the second case), $M\sigma' = k$.
  * In the DDecH', DDecT, DDecT' cases, $\Gamma(M) = \mathrm{seskey}^{l,a}(T''')$ for some l, T'''. Hence, $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma \sim M\sigma' : \mathrm{seskey}^{l,a}(T''') \to c''$ for some c''. Therefore, by Lemma 20 (and since $M\sigma$, $M\sigma'$ are ground), $M\sigma' \in \mathcal{K}$, and $\Gamma(k, M\sigma') <: \mathrm{seskey}^{l,a}(T''')$. Thus, by Lemma 3, $\Gamma(k, M\sigma') = \mathrm{seskey}^{l,a}(T''')$, and since Γ is well-formed, $M\sigma' = k$.

  Let us now show that there exists a ground message N' such that $\sigma'(x) = \mathtt{enc}(N', k)$.
  * In the DDecL, DDecH' and DDecL' cases, $\Gamma(x) = \mathtt{LL}$. Since σ, σ' are well-typed, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{enc}(M', k) \sim \sigma'(x) : \mathtt{LL} \to c''$. Thus by Lemma 17, there exist N, N' such that $\sigma'(x) = \mathtt{enc}(N', N)$. In addition:
    · either $\Gamma_{\mathcal{N},\mathcal{K}} \vdash k \sim N : \mathrm{key}^{\mathtt{HH}}(T''') \to c'''$ for some T''', c''': in that case, by Lemma 20, $N \in \mathcal{K}$ and $\Gamma(k, N) <: \mathrm{key}^{\mathtt{HH}}(T''')$. We have already shown that $\Gamma(k, k)$ is either a subtype of $\mathtt{LL}$, or $\mathrm{seskey}^{l,a}(T'''')$ for some l, T''''. By well-formedness of Γ, only the second case is possible, and it implies that $N = k$.
    · or $\Gamma_{\mathcal{N},\mathcal{K}} \vdash k \sim N : \mathtt{LL} \to c'''$ for some c''': in that case, by Lemma 20, $N = k$.

    In any case we have $\sigma'(x) = \mathtt{enc}(N', k)$.
  * In the DDecT and DDecT' cases, $\Gamma(x) = (T)_{\mathrm{seskey}^{l,a}(T'')}$ and $\Gamma \vdash M \sim M : \mathrm{seskey}^{l,a}(T'') \to \emptyset$ for some l, T''. Hence we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : (T)_{\mathrm{seskey}^{l,a}(T'')} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{enc}(M', k) \sim \sigma'(x) : (T)_{\mathrm{seskey}^{l,a}(T'')} \to c''$. Therefore, by Lemma 17, there exist N, N' such that $\sigma'(x) = \mathtt{enc}(N', N)$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash k \sim N : \mathrm{seskey}^{l,a}(T'') \to \emptyset$. Thus, by Lemma 20, $N \in \mathcal{K}$ and $\Gamma(k, N) = \mathrm{seskey}^{l,a}(T'')$. By well-formedness of Γ, this implies that $N = k$. Therefore $\sigma'(x) = \mathtt{enc}(N', k)$.

  Hence, in any case, $t'\sigma' = \mathtt{dec}(\mathtt{enc}(N', k), k)$. By assumption, σ, σ' are well-typed, and $\sigma(x)\downarrow \neq \bot$. Thus by Lemma 22 $\sigma'(x)\downarrow \neq \bot$. Then $N'\downarrow \neq \bot$. Therefore we have $t'\sigma'\downarrow = N'\downarrow \neq \bot$, which proves the first direction of 1). The other direction is analogous.

- Moreover, still assuming $t\sigma \downarrow \neq \bot$, and keeping the notations from the previous point, we have $t\sigma \downarrow = M'$ and $t'\sigma' \downarrow = N'$. The destructor typing rule applied to prove $\Gamma \vdash_d t \sim t' : T$ can be DDecT, DDecT', DDecL, DDecH', or DDecL'.
  - In the DDecT and DDecT' cases, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : (T)_{\text{seskey}^{l,a}(T'')} \to c''$ for some $c'' \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{enc}(M', k) \sim \mathtt{enc}(N', k) : (T)_{\text{seskey}^{l,a}(T'')} \to c''$. Thus, by Lemma 17, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M' \sim N' : T \to c'''$ for some $c''' \subseteq c''$, and the claim holds.
  - In the DDecL and DDecL' cases, we have $T = \mathtt{LL}$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some $c'' \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{enc}(M', k) \sim \mathtt{enc}(N', k) : \mathtt{LL} \to c''$. In addition we have $\Gamma \vdash M \sim M : \mathtt{LL} \to \emptyset$, and thus $\Gamma \vdash k \sim k : \mathtt{LL} \to c'''$ for some c'''. Therefore, by Lemma 17, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M' \sim N' : \mathtt{LL} \to c'''$ for some $c''' \subseteq c''$ (the case where $\Gamma \vdash k \sim k : \text{key}^{\mathtt{HH}}(T'') \to c'''$ is impossible by Lemma 20, since $\Gamma \vdash k \sim k : \mathtt{LL} \to c'''$), and the claim holds.
  - In the DDecH' case, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some $c'' \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{enc}(M', k) \sim \mathtt{enc}(N', k) : \mathtt{LL} \to c''$. In addition we have $\Gamma \vdash M \sim M : \text{seskey}^{\mathtt{HH},a}(T) \to \emptyset$, and thus $\Gamma \vdash k \sim k : \text{seskey}^{\mathtt{HH},a}(T) \to c'''$ for some c'''. Therefore, by Lemma 17, there exists $c'''' \subseteq c''$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M' \sim N' : T \to c''''$ (the case where $\Gamma \vdash k \sim k : \mathtt{LL} \to c''''$ is impossible by Lemma 20, since $\Gamma \vdash k \sim k : \text{seskey}^{\mathtt{HH},a}(T) \to c'''$), and the claim holds.

  In all cases, point 2) holds, which concludes this case.

– $t = \mathtt{adec}(x, M)$ and $t' = \mathtt{adec}(x, M)$, for some variable $x \in \mathcal{X}$, and some $M \in \mathcal{X} \cup \mathcal{K}$. We know that $\Gamma \vdash_d t \sim t' : T$, which can be proved using the rule DAdecL, DAdecT, DAdecT', DAdecH', or DAdecL'. In the DAdecL, DAdecH', DAdecL' cases, $\Gamma(x) = \mathtt{LL}$, and in the DAdecT and DAdecT' cases $\Gamma(x) = \{T\}_{T'}$ for some T'. By assumption we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \Gamma(x) \to c_x$ for some $c_x \subseteq c$.
  - Let us prove 1) by contraposition. Assume $t\sigma \downarrow \neq \bot$. Hence, $\sigma(x) = \mathtt{aenc}(M', \mathtt{pk}(M\sigma))$ for some M', and $M\sigma$ is a key $k \in \mathcal{K}$. We will now show that $M\sigma' = M\sigma = k$. It is clear if $M \in \mathcal{K}$. Otherwise, $M \in \mathcal{X}$, and:
    - In the DAdecL and DAdecL' cases, $\Gamma \vdash M \sim M : \mathtt{LL} \to \emptyset$. Since σ, σ' are well-typed, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma \sim M\sigma' : \mathtt{LL} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash k \sim M\sigma' : \mathtt{LL} \to c''$. Thus by Lemma 20, either $M\sigma' = k \in \mathcal{FK}$; or $M\sigma' \in \mathcal{BK}$, and $\Gamma(k, M\sigma') <: \text{key}^{\mathtt{LL}}(T''')$ for some T'''. Hence, in any case (by well-formedness of Γ in the case where $M\sigma' \in \mathcal{BK}$), $M\sigma' = k$.
    - In the DAdecH', DAdecT, DAdecT' cases, $\Gamma(M) = \text{seskey}^{l,a}(T''')$ for some l, T'''. Hence, $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma \sim M\sigma' : \text{seskey}^{l,a}(T''') \to c''$ for some c''. Therefore, by Lemma 20 (and since $M\sigma$, $M\sigma'$ are ground), $M\sigma' \in \mathcal{K}$, and $\Gamma(k, M\sigma') <: \text{seskey}^{l,a}(T''')$. Thus, by Lemma 3, $\Gamma(k, M\sigma') = \text{seskey}^{l,a}(T''')$, and since Γ is well-formed, $M\sigma' = k$.

  Let us now show that there exists a ground message N' such that $\sigma'(x) = \mathtt{aenc}(N', \mathtt{pk}(k))$.
    - In the DAdecL, DAdecH' and DAdecL' cases, $\Gamma(x) = \mathtt{LL}$. Since σ, σ' are well-typed, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{aenc}(M', \mathtt{pk}(k)) \sim \sigma'(x) : \mathtt{LL} \to c''$. Thus by Lemma 17, there exist N, N' such that $\sigma'(x) = \mathtt{aenc}(N', N)$. In addition:
      - either $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{pk}(k) \sim N : \text{pkey}(T''') \to c'''$ and $T''' <: \text{key}^{\mathtt{HH}}(T'''')$ for some T''', T'''', c''': in that case, by Lemma 20, $N = \mathtt{pk}(k')$ for some key $k' \in \mathcal{K}$ such that $\Gamma(k, k') <: \text{key}^{\mathtt{HH}}(T''')$. We have already shown that $\Gamma(k, k)$ is either a subtype of $\mathtt{LL}$, or $\text{seskey}^{l,a}(T'''')$ for some l, T''''. By well-formedness of Γ, only the second case is possible, and it implies that $k' = k$.
      - or $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{pk}(k) \sim N : \mathtt{LL} \to c'''$ for some c''': in that case, by Lemma 20, $N = \mathtt{pk}(k)$.

    In any case we have $\sigma'(x) = \mathtt{aenc}(N', \mathtt{pk}(k))$.
    - In the DAdecT and DAdecT' cases, we have $\Gamma(x) = \{T\}_{\text{pkey}(\text{seskey}^{l,a}(T''))}$ as well as $\Gamma \vdash M \sim M : \text{seskey}^{l,a}(T'') \to \emptyset$ for some l, T''. Hence we have

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \{T\}_{\text{pkey}(\text{seskey}^{l,a}(T''))} \to c''$$

for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{aenc}(M', \mathtt{pk}(k)) \sim \sigma'(x) : \{T\}_{\mathrm{pkey}(\mathrm{seskey}^{l,a}(T''))} \to c''$. Therefore, by Lemma 17, there exist N, N', c''' such that $\sigma'(x) = \mathtt{aenc}(N', N)$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{pk}(k) \sim N : \mathrm{pkey}(\mathrm{seskey}^{l,a}(T'')) \to c'''$. Thus, by Lemma 20, $N = \mathtt{pk}(k')$ for some key $k' \in \mathcal{K}$ and $\Gamma(k, k') = \mathrm{seskey}^{l,a}(T'')$. By well-formedness of Γ, this implies that $k' = k$. Therefore $\sigma'(x) = \mathtt{aenc}(N', \mathtt{pk}(k))$.

Hence, in any case, $t'\sigma' = \mathtt{adec}(\mathtt{aenc}(N', \mathtt{pk}(k)), k)$, By assumption, σ, σ' are well-typed, and $\sigma(x) \downarrow\!\neq \bot$. Thus by Lemma 22 $\sigma'(x) \downarrow\!\neq \bot$. Then $N' \downarrow\!\neq \bot$. Therefore we have $t'\sigma' \downarrow\!= N' \downarrow\!\neq \bot$, which proves the first direction of 1). The other direction is analogous.

- Moreover, still assuming $t\sigma \downarrow\!\neq \bot$, and keeping the notations from the previous point, we have $t\sigma \downarrow\!= M'$ and $t'\sigma' \downarrow\!= N'$. The destructor typing rule applied to prove $\Gamma \vdash_d t \sim t' : T$ can be DADECT, DADECT', DADECL, DADECH', or DADECL'.
  - ∗ In the DADECT and DADECT' cases, we have

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \{T\}_{\mathrm{pkey}(\mathrm{seskey}^{l,a}(T''))} \to c''$$

    for some $c'' \subseteq c$, *i.e.*

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{aenc}(M', \mathtt{pk}(k)) \sim \mathtt{aenc}(N', \mathtt{pk}(k)) : \{T\}_{\mathrm{pkey}(\mathrm{seskey}^{l,a}(T''))} \to c''.$$

    Thus, by Lemma 17, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M' \sim N' : T \to c'''$ for some $c''' \subseteq c''$, and the claim holds.
  - ∗ In the DADECL and DADECL' cases, we have $T = \mathtt{LL}$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some $c'' \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{aenc}(M', \mathtt{pk}(k)) \sim \mathtt{aenc}(N', \mathtt{pk}(k)) : \mathtt{LL} \to c''$. In addition we have $\Gamma \vdash M \sim M : \mathtt{LL} \to \emptyset$, and thus $\Gamma \vdash k \sim k : \mathtt{LL} \to c'''$ for some c'''. Therefore, by Lemma 17, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M' \sim N' : \mathtt{LL} \to c''''$ for some $c'''' \subseteq c''$ (the case where $\Gamma \vdash k \sim k : \mathrm{key}^{\mathtt{HH}}(T'') \to c''''$ is impossible by Lemma 20, since we already know that $\Gamma \vdash k \sim k : \mathtt{LL} \to c'''$), and the claim holds.
  - ∗ In the DADECH' case, we have $T = T' \vee \mathtt{LL}$ for some type T'. In addition we know that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c''$ for some $c'' \subseteq c$, *i.e.*

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{aenc}(M', \mathtt{pk}(k)) \sim \mathtt{aenc}(N', \mathtt{pk}(k)) : \mathtt{LL} \to c''.$$

    In addition, $\Gamma \vdash M \sim M : \mathrm{seskey}^{\mathtt{HH},a}(T') \to \emptyset$, and thus $\Gamma \vdash k \sim k : \mathrm{seskey}^{\mathtt{HH},a}(T') \to c'''$ for some c'''. Therefore, by Lemma 17, we know that
    - · either there exist types T'', T''', and constraints $c'''', c''''' \subseteq c''$ such that T'' is a subtype of $\mathrm{key}^{\mathtt{HH}}(T''')$, $\Gamma \vdash \mathtt{pk}(k) \sim \mathtt{pk}(k) : \mathrm{pkey}(T'') \to c''''$, and $\Gamma \vdash M' \sim N' : T''' \to c''''$. Since $\Gamma \vdash \mathtt{pk}(k) \sim \mathtt{pk}(k) : \mathrm{pkey}(T'') \to \emptyset$, by Lemma 20, we have $\Gamma(k, k) <: \mathrm{key}^{\mathtt{HH}}(T''')$. As we already know that $\Gamma \vdash k \sim k : \mathrm{seskey}^{\mathtt{HH},a}(T') \to c'''$, by the same lemma and Lemma 3, we have $T''' = T'$. Thus $\Gamma \vdash M' \sim N' : T' \to c''''$, and by rule TOR, we have $\Gamma \vdash M' \sim N' : T' \vee \mathtt{LL} \to c''''$.
    - · or $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c''$, and by rule TOR we have $\Gamma \vdash M' \sim N' : T' \vee \mathtt{LL} \to c''''$.

    In all cases, $\Gamma \vdash M' \sim N' : T \to c''''$ for some $c'''' \subset c''$, and point 2) holds, which concludes this case.

– $t = t' = \mathtt{checksign}(x, M)$. We know that $\Gamma \vdash_d t \sim t' : T$, which can be proved using either DCHECKH, DCHECKH', DCHECKL, or DCHECKL'. In both cases, $\Gamma(x) = \mathtt{LL}$. M is either a verification key or a variable. By assumption we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c_x$ for some $c_x \subseteq c$.

- Let us prove 1) by contraposition. Assume $t\sigma \downarrow\!\neq \bot$. Hence, $\sigma(x) = \mathtt{sign}(M', M''\sigma)$ for some M', M'', and $M''\sigma$ is a key $k \in \mathcal{K}$ such that $M\sigma = \mathtt{vk}(k)$.

We will now show that $M\sigma' = M\sigma = \mathtt{vk}(k)$. It is clear if M is a verification key. Otherwise, $M \in \mathcal{X}$, which means the rule applied to prove $\Gamma \vdash_d t \sim t' : T$ is DCHECKL' or DCHECKH'. In either case, from the form of the rule we have $\Gamma \vdash M \sim M : \mathtt{LL} \to \emptyset$, Since σ, σ' are well-typed, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma \sim M\sigma' : \mathtt{LL} \to c''$ for some c'', *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{vk}(k) \sim M\sigma' : \mathtt{LL} \to c''$. Thus by Lemma 20, and since Γ is well-formed, $M\sigma' = \mathtt{vk}(k)$.
In addition, we know that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c_x$, *i.e.*

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{sign}(M', k) \sim \sigma'(x) : \mathtt{LL} \to c_x.$$

Hence Lemma 18 guarantees that there exist N', N'' such that $\sigma'(x) = \mathtt{sign}(N', N'')$, and either $\Gamma \vdash k \sim N'' : \mathrm{eqkey}^{\mathtt{HH}}(T') \to \emptyset$ for some T' or $\Gamma \vdash k \sim N'' : \mathtt{LL} \to c''$ for some c''. In both cases, Lemma 20 implies that $N'' = k$. Thus we have $\sigma'(x) = \mathtt{sign}(N', k)$.
Hence, $t'\sigma' = \mathtt{checksign}(\mathtt{sign}(N', k), \mathtt{vk}(k))$, By assumption, σ, σ' are well-typed, and $\sigma(x) \downarrow\neq \bot$. Thus by Lemma 22 $\sigma'(x) \downarrow\neq \bot$. Then $N' \downarrow\neq \bot$. Therefore we have $t'\sigma' \downarrow= N' \downarrow\neq \bot$, which proves the first direction of 1). The other direction is analogous.

- Moreover, still assuming $t\sigma \downarrow\neq \bot$, and keeping the notations from the previous point, we have $t\sigma \downarrow= M'$ and $t'\sigma' \downarrow= N'$. The destructor typing rule applied to prove $\Gamma \vdash_d t \sim t' : T$ can be DCHECKH, DCHECKH', DCHECKL, or DCHECKL'.
  - ∗ In the DCHECKH and DCHECKH' cases we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c_x$ for $c_x \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{sign}(M', k) \sim \mathtt{sign}(N', k) : \mathtt{LL} \to c_x$. In both cases we have $\Gamma \vdash \mathtt{vk}(k) \sim \mathtt{vk}(k) : \mathrm{vkey}(T') \to c''$ for some c'' and some $T' <: \mathrm{key}^{\mathtt{HH}}(T)$. Hence by Lemma 20, $\Gamma(k, k) <: \mathrm{key}^{\mathtt{HH}}(T)$. By Lemma 18, we know that there exist $c' \subseteq c_x$ such that $\Gamma \vdash M' \sim N' : T \to c'$ (the case where $\Gamma \vdash k \sim k : \mathtt{LL} \to c''$ is impossible by Lemma 20 since $\Gamma(k, k) <: \mathrm{key}^{\mathtt{HH}}(T)$ and Γ is well-formed). This proves point 2).
  - ∗ In the DCHECKL and DCHECKL' cases we have $T = \mathtt{LL}$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \mathtt{LL} \to c_x$ for $c_x \subseteq c$, *i.e.* $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \mathtt{sign}(M', k) \sim \mathtt{sign}(N', k) : \mathtt{LL} \to c_x$. By Lemma 18, we know that there exist $c' \subseteq c_x$ such that $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'$. This proves point 2).

  In all cases, point 2) holds, which concludes this case.

– $t = t' = \pi_1(x)$. We know that $\Gamma \vdash_d t : t'T$, which can be proved using either rule DFST or DFSTL. In the first case, $\Gamma(x) = T_1 * T_2$ is a pair type, and in the second case $\Gamma(x) = \mathtt{LL}$.

- We prove 1) by contraposition. Assume $t\sigma \downarrow\neq \bot$. Hence, $\sigma(x) = \langle M_1, M_2 \rangle$ for some M_1, M_2. By assumption σ, σ' are well-typed. Thus $\Gamma \vdash \sigma(x) \sim \sigma'(x) : \Gamma(x) \to c_x$, for some $c_x \subseteq c$. Thus, by applying Lemma 19, in any case we know that there exist N_1, N_2 such that $N = \langle N_1, N_2 \rangle$. Since, σ, σ' are well-typed, and $\sigma(x) \downarrow\neq \bot$, by Lemma 22, $\sigma'(x) \downarrow\neq \bot$. Then $N_1 \downarrow\neq \bot$. Therefore we have $t'\sigma' \downarrow= N_1 \downarrow\neq \bot$, which proves the first direction of 1). The other direction is analogous.
- Moreover, still assuming $t\sigma \downarrow\neq \bot$, and keeping the notations from the previous point, we have $t\sigma \downarrow= M_1$ and $t'\sigma' \downarrow= N_1$. In addition, we know that $\Gamma \vdash_d t : t'T$, which can be proved using either rule DFST or DFSTL. Lemma 19, which we applied in the previous point, also implies that there exist c_1, c_2, such that $c = c_1 \cup c_2$ and for $i \in \{1, 2\}$, $\Gamma \vdash M_i \sim N_i : T_i \to c_i$ (in the DFST case) or $\Gamma \vdash M_i \sim N_i : \mathtt{LL} \to c_i$ (in the DFSTL case).
  We distinguish two cases for the rule applied to prove $\Gamma \vdash_d t : t'T$.
  - ∗ DFST: Then $\Gamma(x) = T * T_2$ for some T_2, and $\Gamma \vdash M_1 \sim N_1 : T \to c_1 (\subseteq c)$ proves 2).
  - ∗ DFSTL: Then $T = \Gamma(x) = \mathtt{LL}$, and $\Gamma \vdash M_1 \sim N_1 : \mathtt{LL} \to c_1 (\subseteq c)$ proves 2).

  In both cases, point 2) holds, which concludes this case.

– $t = t' = \pi_2(x)$. This case is similar to the previous one.

Lemma 22 (Typable messages either reduce on both sides, or fail on both sides). *For all (well-formed) Γ, for all messages M, M', for all T, c, if*

$$\Gamma \vdash M \sim N : T \to c,$$

then

$$M \downarrow = \bot \iff N \downarrow = \bot.$$

Proof. Note that since messages do not contain destructors, it is clear that for any message M, either $M \downarrow = M$ or $M \downarrow = \bot$.

We prove the lemma by induction on the type derivation for $\Gamma \vdash M \sim N : T \to c$.

In the TNONCE, TNONCEL, TCSTFN, TPUBKEYL, TVKEYL, TKEY, TVAR, TLR^1, TLR$^\infty$, TLRVAR, it is clear that $M \downarrow = M \neq \bot$ and $N \downarrow = N \neq \bot$.

In the THIGH case, the premise of the rule implies that $M \downarrow \neq \bot$ and $N \downarrow \neq \bot$.

In the TLR' and TLRL' cases, there exist l, l', a, m, n, c such that $\Gamma \vdash M \sim N : [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!] \to c$. Hence by Lemma 16, either M, N are variables, or $M = m$ and $N = n$. Either way, $M \downarrow = M \neq \bot$ and $N \downarrow = N \neq \bot$.

The TENC case is more involved. In that case, there exist M', M'', N', N'', T', T'' such that $M = \mathtt{enc}(M', M'')$, $N = \mathtt{enc}(N', N'')$, and $T = (T')_{T''}$. Let us show that $M \downarrow \neq \bot \Rightarrow N \downarrow \neq \bot$ (the other direction has a similar proof). Since $M \downarrow \neq \bot$, we have $M' \downarrow \neq \bot$. Hence, as $\Gamma \vdash M' \sim N' : T' \to c'$ for some c', by the induction hypothesis, $N' \downarrow \neq \bot$. Since $M \downarrow \neq \bot$, we also have $M'' \downarrow \in \mathcal{K}$, *i.e.* by the previous remark, $M'' \in \mathcal{K}$. In addition, $\Gamma \vdash M'' \sim N'' : T'' \to c''$ for some c'', and either $T'' = \mathtt{LL}$ or $\exists l, T'''.T'' <: \mathrm{key}^l(T''')$. Hence, by Lemma 20, either $M'', N'' \in \mathcal{K}$, or $M'', N'' \in \mathcal{X}$. The second case is impossible since $M'' \in \mathcal{K}$. We thus have $N' \downarrow \neq \bot$ and $N'' \downarrow \in \mathcal{K}$. Therefore, $N \downarrow \neq \bot$ and the claim holds.

The TAENC, TPUBKEY, TVKEY, TSIGNH, TSIGNL cases are similar to the TENC case.

In the remaining cases, *i.e.* TPAIR, TENCH, TENCL, TAENCH, TAENCL, THASH, THASHL, TSUB, and TOR, the claim directly follows from the induction hypothesis. We write the proof for the TPAIR case. In that case there exist M', M'', N', N'', T', T'' such that $M = \langle M', M'' \rangle$, $N = \langle N', N'' \rangle$, and $T = T' * T''$. Since $\Gamma \vdash M' \sim N' : T' \to c'$ for some c', by the induction hypothesis, $M' \downarrow = \bot \iff N' \downarrow = \bot$. Similarly, $M'' \downarrow = \bot \iff N'' \downarrow = \bot$. Therefore, $M \downarrow = \bot \iff N \downarrow = \bot$, and the claim holds.

Lemma 23 (LL type is preserved by attacker terms). *For all (well-formed) Γ, for all frames ϕ and ϕ' with $\Gamma \vdash \phi \sim \phi' : \mathtt{LL} \to c$, for all attacker term R such that* $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi)$,
either there exists $c' \subseteq c$ such that

$$\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \mathtt{LL} \to c'$$

or

$$R\phi \downarrow = R\phi' \downarrow = \bot.$$

Proof. We show this property by induction over the attacker term R.

Induction Hypothesis: the statement holds for all subterms of R.

There are several cases for R. The base cases are the cases where R is a variable, a name in $\mathcal{FN}$ or a constant in $\mathcal{C}$.

1. $\underline{R = x}$ Since $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi)$, we have $x \in \mathrm{dom}(\phi) = \mathrm{dom}(\phi')$, hence $R\phi = \phi(x)$. By assumption, there exists $c' \subseteq c$ such that $\Gamma \vdash \phi(x) \sim \phi'(x) : \mathtt{LL} \to c'$. Thus, by Lemma 22, either $\phi(x) = \phi'(x) = \bot$, or $\phi(x) \downarrow = \phi(x)$ and $\phi'(x) \downarrow = \phi'(x)$. In the first case the claim clearly holds. In the second case, $R\phi \downarrow = \phi(x)$ and $R\phi' \downarrow = \phi'(x)$. Since $\Gamma \vdash \phi(x) \sim \phi'(x) : \mathtt{LL} \to c' \subseteq c$, the claim also holds.

2. $\underline{R = a}$ with $a \in \mathcal{C} \cup \mathcal{FN} \cup \mathcal{FK}$. Then $R\phi \downarrow = R\phi' \downarrow = a$ and by rule TCSTFN, we have $\Gamma \vdash a \sim a : \mathtt{LL} \to \emptyset$. Hence the claim holds.
3. $\underline{R = \mathtt{pk}(K)}$ We apply the induction hypothesis to K and distinguish three cases.
   (a) $\underline{\text{If } K\phi \downarrow = \bot}$ then $K\phi' \downarrow = \bot$, hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (b) $\underline{\text{If } K\phi \downarrow \neq \bot \text{ and is not a key}}$ then $K\phi' \downarrow \neq \bot$ (by IH), and by IH we have $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$ for some $c' \subseteq c$. Then by Lemma 20, $K\phi' \downarrow$ is not a key either. Hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (c) $\underline{\text{If } K\phi \downarrow \text{ is a key}}$, then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$. Hence by Lemma 20 (and since Γ is well-formed) $K\phi' \downarrow = K\phi' \downarrow \in \mathcal{K}$, and either $\Gamma(K\phi \downarrow, K\phi' \downarrow) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, or $K\phi' \downarrow \in \mathcal{FK}$. Therefore by rule TPUBKEYL, $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \mathtt{LL} \to \emptyset$, and the claim holds.
4. $\underline{R = \mathtt{vk}(K)}$ This case is analogous to the pk case.
5. $\underline{R = \langle R_1, R_2 \rangle}$ where R_1 and R_2 are also attacker terms. We then apply the induction hypothesis to the same frames and R_1, R_2. We distinguish two cases:
   (a) $\underline{R_1\phi \downarrow = \bot \vee R_2\phi \downarrow = \bot}$ In this case we also have $R_1\phi' \downarrow = \bot \vee R_2\phi' \downarrow = \bot$ and therefore $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (b) $\underline{R_1\phi \downarrow \neq \bot \wedge R_2\phi \downarrow \neq \bot}$ In this case, by the induction hypothesis, we also have $R_1\phi' \downarrow \neq \bot \wedge R_2\phi' \downarrow \neq \bot$, and we also know that there exist $c_1 \subseteq c$ and $c_2 \subseteq c$ such that $\Gamma \vdash R_1\phi \downarrow \sim R_1\phi' \downarrow : \mathtt{LL} \to c_1$ and $\Gamma \vdash R_2\phi \downarrow \sim R_2\phi' \downarrow : \mathtt{LL} \to c_2$.
   Thus, by the rule TPAIR followed by TSUB, $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \mathtt{LL} \to c_1 \cup c_2$. Since $c_1 \cup c_2 \subseteq c$, this proves the case.
6. $\underline{R = \mathtt{enc}(S, K)}$ We apply the induction hypothesis to K and distinguish three cases.
   (a) $\underline{\text{If } K\phi \downarrow = \bot}$ then $K\phi' \downarrow = \bot$, hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (b) $\underline{\text{If } K\phi \downarrow \neq \bot \text{ and is not a key}}$ then $K\phi' \downarrow \neq \bot$ (by IH), and by IH we have $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$ for some $c' \subseteq c$. Then, by Lemma 20, $K\phi' \downarrow$ is not a key either. Hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (c) $\underline{\text{If } K\phi \downarrow \text{ is a key}}$, then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$. Hence, by Lemma 20 (and since Γ is well-formed), $K\phi' \downarrow = K\phi \downarrow \in \mathcal{K}$, and either $\Gamma(K\phi \downarrow, K\phi' \downarrow) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, or $K\phi \downarrow \in \mathcal{FK}$. Thus, in each case, by rules TKEY and TSUB or TCSTFN, $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to \emptyset$. We then apply the IH to S, and either $S\phi \downarrow = S\phi' \downarrow = \bot$, in which case $R\phi \downarrow = R\phi' \downarrow = \bot$; or there exists $c'' \subseteq c$ such that $\Gamma \vdash S\phi \downarrow \sim S\phi' \downarrow : \mathtt{LL} \to c''$. Since $R\phi \downarrow = \mathtt{enc}(S\phi \downarrow, K\phi \downarrow)$, and similarly for ϕ', by rule TENC, we have $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : (\mathtt{LL})_{\mathtt{LL}} \to c''$, and then by rule TENCL, $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \mathtt{LL} \to c''$.
7. $\underline{R = \mathtt{aenc}(S, K)}$ We apply the induction hypothesis to K and distinguish three cases.
   (a) $\underline{\text{If } K\phi \downarrow = \bot}$ then $K\phi' \downarrow = \bot$, hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (b) $\underline{\text{If } K\phi \downarrow \neq \bot \text{ and is not } \mathtt{pk}(k) \text{ for some } k \in \mathcal{K}}$ then $K\phi' \downarrow \neq \bot$ (by IH), and by IH there exists $c' \subseteq c$ such that $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$. Then, by Lemma 20, $K\phi' \downarrow$ is not a public key either. Hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (c) $\underline{\text{If } K\phi \downarrow = \mathtt{pk}(k) \text{ for some } k \in \mathcal{K}}$, then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash \mathtt{pk}(k) \sim K\phi' \downarrow : \mathtt{LL} \to c'$. Thus, by Lemma 20, $K\phi' \downarrow = \mathtt{pk}(N)$ for some N such that either $N = k \in \mathrm{keys}(\Gamma) \cup \mathcal{FK}$; or $\Gamma \vdash k \sim N : \mathrm{eqkey}^{l}(T) \to c''$ for some l, T, c''. In the second case, the same lemma and the well-formedness of Γ also imply that $N = k$ and $k \in \mathrm{keys}(\Gamma)$. Thus, by rule TPUBKEYL, $\Gamma \vdash \mathtt{pk}(k) \sim \mathtt{pk}(k) : \mathtt{LL} \to \emptyset$. We then apply the IH to S, and either $S\phi \downarrow = S\phi' \downarrow = \bot$, in which case $R\phi \downarrow = R\phi' \downarrow = \bot$; or there exists $c''' \subseteq c$ such that $\Gamma \vdash S\phi \downarrow \sim S\phi' \downarrow : \mathtt{LL} \to c'''$. Therefore, by rule TAENC, $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \{\mathtt{LL}\}_{\mathtt{LL}} \to c''$, and by rule TAENCL we have $\Gamma \vdash R\phi \downarrow \sim R\phi' \downarrow : \mathtt{LL} \to c''$.
8. $\underline{R = \mathtt{sign}(S, K)}$ We apply the induction hypothesis to K and distinguish three cases.
   (a) $\underline{\text{If } K\phi \downarrow = \bot}$ then $K\phi' \downarrow = \bot$, hence $R\phi \downarrow = R\phi' \downarrow = \bot$.
   (b) $\underline{\text{If } K\phi \downarrow \neq \bot \text{ and is not a key } k \in \mathcal{K}}$ then $K\phi' \downarrow \neq \bot$ (by IH), and by IH we have $\Gamma \vdash K\phi \downarrow \sim K\phi' \downarrow : \mathtt{LL} \to c'$ for some $c' \subseteq c$. Then, by Lemma 20, $K\phi' \downarrow$ is not a key either. Hence $R\phi \downarrow = R\phi' \downarrow = \bot$.

(c) If $K\phi\downarrow = k$ for some $k \in \mathcal{K}$, then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash K\phi\downarrow \sim K\phi'\downarrow : \mathtt{LL} \to c'$. Hence by Lemma 20, (and since Γ is well-formed) $K\phi'\downarrow = k \in \mathcal{K}$ and either $\Gamma(k,k) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, or $k \in \mathcal{FK}$. Thus, in either case, by rules TKEY, TSUB, and TCSTFN, $\Gamma \vdash K\phi\downarrow \sim K\phi'\downarrow : \mathtt{LL} \to \emptyset$. We then apply the IH to S, and either $S\phi\downarrow = S\phi'\downarrow = \bot$, in which case $R\phi\downarrow = R\phi'\downarrow = \bot$; or there exists $c'' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c''$. Therefore by rule TSIGNL, $\Gamma \vdash R\phi\downarrow \sim R\phi'\downarrow : \mathtt{LL} \to c''$.

9. $R = \mathtt{h}(S)$ We apply the induction hypothesis to S. We distinguish two cases:
   (a) $S\phi\downarrow = \bot$ In this case we also have $S\phi'\downarrow = \bot$ and therefore $R\phi\downarrow = R\phi'\downarrow = \bot$.
   (b) $S\phi\downarrow \neq \bot$ In this case, by the induction hypothesis, we also have $S\phi'\downarrow \neq \bot$, and we also know that there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c'$ Thus, by rule THASHL, $\Gamma \vdash R\phi\downarrow \sim R\phi'\downarrow : \mathtt{LL} \to c'$, which proves this case.
10. $R = \pi_1(S)$ We apply the induction hypothesis to S and distinguish three cases.
   (a) $S\phi\downarrow = \bot$ Then $S\phi'\downarrow = \bot$ (by IH), hence $R\phi\downarrow = R\phi'\downarrow = \bot$.
   (b) $S\phi\downarrow \neq \bot$ and is not a pair Then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c'$, which implies that $R\phi'\downarrow$ and is not a pair either by Lemma 19. Hence $R\phi\downarrow = R\phi'\downarrow = \bot$.
   (c) $S\phi\downarrow = \langle M_1, M_2\rangle$ is a pair Then by IH there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c'$. This implies, by Lemma 19, that $S\phi'\downarrow = \langle M_1', M_2'\rangle$ is also a pair, and that $\Gamma \vdash M_1 \sim M_1' : \mathtt{LL} \to c''$ for some $c'' \subseteq c'$. Since $R\phi\downarrow = M_1$ and $R\phi'\downarrow = M_1'$, this proves the case.
11. $R = \pi_2(S)$ This case is analogous to the case 10.
12. $R = \mathtt{dec}(S, K)$ We apply the induction hypothesis to K and, similarly to the case 6, we distinguish several cases.
   (a) If $K\phi\downarrow = \bot$ or is not a key then, as in case 6, $R\phi\downarrow = R\phi'\downarrow = \bot$.
   (b) If $K\phi\downarrow$ is a key, then similarly to case 6 we can show that $K\phi\downarrow = K\phi'\downarrow$, and either $\Gamma(K\phi\downarrow, K\phi'\downarrow) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, or $K\phi\downarrow \in \mathcal{FK}$. We then apply the IH to S, which creates two cases. Either $S\phi\downarrow = S\phi'\downarrow = \bot$, or there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c'$. In the first case, the claim holds, since $R\phi\downarrow = R\phi'\downarrow = \bot$. In the second case, by Lemmas 17 and 20, and since Γ is well-formed, we know that $S\phi\downarrow$ is a message encrypted with $K\phi\downarrow$ if and only if $S\phi'\downarrow$ also is an encryption by this key. Consequently, if $S\phi\downarrow$ is not an encryption by $K\phi\downarrow$ $(= K\phi'\downarrow)$, then it is the same for $S\phi'\downarrow$; and $R\phi\downarrow = R\phi'\downarrow = \bot$. Otherwise, $S\phi\downarrow = \mathtt{enc}(M, K\phi\downarrow)$ and $S\phi'\downarrow = \mathtt{enc}(N, K\phi'\downarrow)$ for some M, N. In that case, by IH, we have $\Gamma \vdash \mathtt{enc}(M, K\phi\downarrow) \sim \mathtt{enc}(N, K\phi'\downarrow) : \mathtt{LL} \to c'$. Therefore, by Lemma 17, $\Gamma \vdash M \sim N : \mathtt{LL} \to c'$, which is to say $\Gamma \vdash R\phi\downarrow \sim R\phi'\downarrow : \mathtt{LL} \to c'$. Hence the claim holds in this case.
13. $R = \mathtt{adec}(S, K)$ We apply the induction hypothesis to K and, similarly to the case 6, we distinguish several cases.
   (a) If $K\phi\downarrow = \bot$ or is not a key then, as in case 6, $R\phi\downarrow = R\phi'\downarrow = \bot$.
   (b) If $K\phi\downarrow$ is a key, then similarly to case 6 we can show that $K\phi\downarrow = K\phi'\downarrow$, and either $\Gamma(K\phi\downarrow, K\phi'\downarrow) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some T, or $K\phi\downarrow \in \mathcal{FK}$. We then apply the IH to S, which creates two cases. Either $S\phi\downarrow = S\phi'\downarrow = \bot$, or there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \mathtt{LL} \to c'$. In the first case, the claim holds, since $R\phi\downarrow = R\phi'\downarrow = \bot$. In the second case, by Lemmas 17 and 20, and since Γ is well-formed, we know that $S\phi\downarrow$ is a message asymmetrically encrypted by $\mathtt{pk}(K\phi\downarrow)$ if and only if $S\phi'\downarrow$ also is an asymmetric encryption by this key. Consequently, if $S\phi\downarrow$ is not an encryption by $\mathtt{pk}(K\phi\downarrow)$ $(= \mathtt{pk}(K\phi'\downarrow))$, then it is the same for $S\phi'\downarrow$, and $R\phi\downarrow = R\phi'\downarrow = \bot$. Otherwise, $S\phi\downarrow = \mathtt{aenc}(M, \mathtt{pk}(K\phi\downarrow))$ and $S\phi'\downarrow = \mathtt{aenc}(N, \mathtt{pk}(K\phi'\downarrow))$ for some M, N. Thus by IH we have $\Gamma \vdash \mathtt{aenc}(M, \mathtt{pk}(K\phi\downarrow)) \sim \mathtt{aenc}(N, \mathtt{pk}(K\phi'\downarrow)) : \mathtt{LL} \to c'$. Therefore, by Lemma 17 (point 6), we know that $\Gamma \vdash M \sim N : \mathtt{LL} \to c'$, which is to say $\Gamma \vdash R\phi\downarrow \sim R\phi'\downarrow : \mathtt{LL} \to c'$. Hence the claim holds in this case.

14. $R = \texttt{checksign}(S, K)$ We apply the induction hypothesis to K and, similarly to the case 7, we distinguish several cases.
    (a) If $K\phi\downarrow = \bot$ or is not a verification key then, as in case 7, we can show that $R\phi\downarrow = R\phi'\downarrow = \bot$.
    (b) If $K\phi\downarrow$ is a verification key $\texttt{vk}(k)$ for some $k \in \mathcal{K}$, then similarly to case 7 we can show that $K\phi\downarrow = K\phi'\downarrow$, and $k \in \text{keys}(\Gamma) \cup \mathcal{FK}$. We then apply the IH to S, which creates two cases. Either $S\phi\downarrow = S\phi'\downarrow = \bot$, or there exists $c' \subseteq c$ such that $\Gamma \vdash S\phi\downarrow \sim S\phi'\downarrow : \texttt{LL} \to c'$. In the first case, the claim holds, since $R\phi\downarrow = R\phi'\downarrow = \bot$. In the second case, by Lemmas 18 and 20, and since Γ is well-formed, we know that $S\phi\downarrow$ is a signature by $k (= K\phi\downarrow)$ if and only if $S\phi'\downarrow$ also is a signature by this key. Consequently, if $S\phi\downarrow$ is not signed by k, then neither is $S\phi'\downarrow$, and $R\phi\downarrow = R\phi'\downarrow = \bot$. Otherwise, $S\phi\downarrow = \texttt{sign}(M, k)$ and $S\phi'\downarrow = \texttt{sign}(N, k)$ for some M, N. Thus by IH we have $\Gamma \vdash \texttt{sign}(t, k) \sim \texttt{sign}(t', k) : \texttt{LL} \to c'$. Therefore, by Lemma 18 (point 2), we know that there exists $c'' \subseteq c'$ such that $\Gamma \vdash M \sim N : \texttt{LL} \to c''$. That is to say $\Gamma \vdash R\phi\downarrow \sim R\phi'\downarrow : \texttt{LL} \to c''$. Hence the claim holds in this case.

Lemma 24 (Substitution preserves typing). *For all Γ, Γ', such that $\Gamma \cup \Gamma' \vdash \diamond$, (we do not require that Γ' is well-formed), for all M, N, T, c_σ, c, for all ground substitutions σ, σ', if*

- *Γ' only contains variables;*
- *Γ and Γ' have disjoint domains;*
- *for all $x \in \text{dom}(\Gamma)$, $\Gamma(x)$ is not of the form $[\![\tau_m^{l,1}; \tau_n^{l',1}]\!]$,*
- *for all $x \in \text{dom}(\sigma)$, $\sigma(x)\downarrow = \sigma(x)$, and similarly for σ',*
- *$(\Gamma_{\mathcal{N},\mathcal{K}} \cup \Gamma')_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : (\Gamma_{\mathcal{N},\mathcal{K}} \cup \Gamma')_{\mathcal{X}} \to c_\sigma$,*
- *and $\Gamma \cup \Gamma' \vdash M \sim N : T \to c$*

then there exists c' such that $[\![c]\!]_{\sigma,\sigma'} \subseteq c' \subseteq [\![c]\!]_{\sigma,\sigma'} \cup c_\sigma$ and

$$\Gamma \vdash M\sigma \sim N\sigma' : T \to c'.$$

In particular, if we have $\Gamma' = {\Gamma'''}_{\mathcal{X}}$ and $\Gamma = {\Gamma'''}_{\mathcal{N},\mathcal{K}}$ for some Γ''', then the first three conditions trivially hold.

Proof. Note that ${\Gamma'}_{\mathcal{N},\mathcal{K}} = \emptyset$, and ${\Gamma'}_{\mathcal{X}} = \Gamma'$.

This proof is done by induction on the typing derivation for the terms. The claim clearly holds in the TNONCE, TNONCEL, TCSTFN, TPUBKEYL, TVKEYL, TKEY, TLR^1, TLR$^\infty$ since their conditions do not use $\Gamma(x)$ (for any variable x) or another type judgement, and they still apply to the messages $M\sigma$ and $N\sigma'$.

In the THIGH case, we have

$$\text{names}(\sigma) \cup \text{names}(\sigma') \cup \text{vars}(\sigma) \cup \text{vars}(\sigma') \cup \text{keys}(\sigma) \cup \text{keys}(\sigma') \subseteq \text{dom}(\Gamma) \cup \text{keys}(\Gamma) \cup \mathcal{FN} \cup \mathcal{FK},$$

and

$$\text{names}(M) \cup \text{names}(N) \cup \text{vars}(M) \cup \text{vars}(N) \cup \text{keys}(M) \cup \text{keys}(N) \subseteq \text{dom}(\Gamma) \cup \text{dom}(\sigma) \cup \text{keys}(\Gamma) \cup \mathcal{FN} \cup \mathcal{FK},$$

therefore

$$\text{names}(M\sigma) \cup \text{names}(N\sigma') \cup \text{vars}(M\sigma) \cup \text{vars}(N\sigma') \cup \text{keys}(M\sigma) \cup \text{keys}(N\sigma') \subseteq \text{dom}(\Gamma) \cup \text{keys}(\Gamma) \cup \mathcal{FN} \cup \mathcal{FK}.$$

In addition, since $M \downarrow \neq \bot$, and $\forall x \in \text{dom}(\sigma), \sigma(x)\downarrow = \sigma(x)$, we have $M\sigma \downarrow \neq \bot$. Similarly, $N\sigma' \downarrow \neq \bot$. Therefore, rule THIGH may be applied to obtain $\Gamma \vdash M\sigma \sim N\sigma' : \texttt{HL} \to \emptyset$.

The claim follows directly from the induction hypothesis in all other cases except the TVAR and TLRVAR cases, which are the base cases.

In the TVAR case, the claim also holds, since $M = N = x$ for some variable $x \in \mathrm{dom}(\Gamma) \cup \mathrm{dom}(\Gamma')$. If $x \in \mathrm{dom}(\Gamma)$, then $x\sigma = x\sigma' = x$, and $T = \Gamma(x)$. Thus, by rule TVAR, $\Gamma \cup \Gamma' \vdash x\sigma \sim x\sigma' : \Gamma(x) \to \emptyset$ and the claim holds. If $x \in \mathrm{dom}(\Gamma')$, then $T = \Gamma'(x)$, and, since by hypothesis the substitutions are well-typed, there exists $c_x \subseteq c_{\sigma,\sigma'}$ such that $(\Gamma \cup \Gamma')_{\mathcal{N},\mathcal{K}} \vdash \sigma(x) \sim \sigma'(x) : \Gamma'(x) \to c_x$. Thus, since $(\Gamma \cup \Gamma'')_{\mathcal{N},\mathcal{K}} = \Gamma_{\mathcal{N},\mathcal{K}}$, and by applying Lemma 12 to Γ, $\Gamma \vdash \sigma(x) \sim \sigma'(x) : \Gamma'(x) \to c_x$ and the claim holds.

Finally, in the TLRVAR case, there exist two variables x, y, and types $\tau_m^{l,1}$, $\tau_n^{l',1}$, $\tau_{m'}^{l'',1}$, $\tau_{n'}^{l''',1}$, such that $M = x$, $N = y$, $c = \emptyset$, $\Gamma \cup \Gamma' \vdash x \sim x : [\![\tau_m^{l,1} ; \tau_n^{l',1}]\!] \to \emptyset$, $\Gamma \cup \Gamma' \vdash y \sim y : [\![\tau_{m'}^{l'',1} ; \tau_{n'}^{l''',1}]\!] \to \emptyset$, and $T = [\![\tau_m^{l,1} ; \tau_{n'}^{l''',1}]\!]$.

By Lemma 16, this implies that $(\Gamma \cup \Gamma')(x) = [\![\tau_m^{l,1} ; \tau_n^{l',1}]\!]$ and $(\Gamma \cup \Gamma')(y) = [\![\ l'' ;\ 1]\!] m' l''' 1 n'$. Hence, since by hypothesis Γ does not contain such types, $x \in \mathrm{dom}(\Gamma')$ and $y \in \mathrm{dom}(\Gamma')$.

Moreover, by the induction hypothesis, there exist c', $c'' \subseteq c_\sigma$ such that $\Gamma \vdash x\sigma \sim x\sigma' : [\![\tau_m^{l,1} ; \tau_n^{l',1}]\!] \to c'$, and $\Gamma \vdash y\sigma \sim y\sigma' : [\![\tau_{m'}^{l'',1} ; \tau_{n'}^{l''',1}]\!] \to c''$. That is to say, since $x, y \in \mathrm{dom}(\Gamma') = \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma')$, that $\Gamma \vdash \sigma(x) \sim \sigma'(x) : [\![\tau_m^{l,1} ; \tau_n^{l',1}]\!] \to c'$, and $\Gamma \vdash \sigma(y) \sim \sigma'(y) : [\![\tau_{m'}^{l'',1} ; \tau_{n'}^{l''',1}]\!] \to c''$. Hence, by Lemma 16, and since σ, σ' are ground, we have $\sigma(x) = m$, $\sigma'(x) = n$, $\sigma(y) = m'$, and $\sigma'(y) = n'$, and $\Gamma(m) = \tau_m^{l,1}$ and $\Gamma(n') = \tau_{n'}^{l''',1}$.

Thus, by rule TLR[1], $\Gamma \vdash \sigma(x) \sim \sigma'(y) : [\![\tau_m^{l,1} ; \tau_{n'}^{l''',1}]\!] \to \emptyset$, which proves the claim.

Lemma 25 (Types LL and HH are disjoint). *For all well-formed Γ, for all ground terms M, M', N, N', for all sets of constraints c, c', if $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma \vdash M' \sim N' : \mathtt{HH} \to c'$ then $M \neq M'$ and $N \neq N'$.*

Proof. First, it is easy to see by induction on the type derivation that for all ground terms M, N, for all c, if $\Gamma \vdash M \sim N : \mathtt{HH} \to c$ then either

- M is a nonce $m \in \mathcal{N}$ such that $\Gamma(m) = \tau_m^{\mathtt{HH},a}$ for some $a \in \{\infty, 1\}$;
- or M is a key and there exist $k \in \mathcal{K}$ and T such that $\Gamma(M, k) <: \mathrm{key}^{\mathtt{HH}}(T)$;
- or $\Gamma \vdash M \sim N : \mathtt{HH} * T \to c'$ for some T, c';
- or $\Gamma \vdash M \sim N : T * \mathtt{HH} \to c'$ for some T, c'.

Indeed, (as Γ is well-formed) the only possible cases are TNONCE, TSUB, and TLR'. In the TNONCE case the claim clearly holds. In the TSUB case we use Lemma 3 followed by Lemma 20. In the TLR' case we apply Lemma 16 and the claim directly follows.

Let us now show that for all M, N, N' ground, for all c, c', $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma \vdash M \sim N' : \mathtt{HH} \to c'$ cannot both hold. (This corresponds, with the notations of the statement of the lemma, to proving by contradiction that $M \neq M'$. The proof that $N \neq N'$ is analogous.)

We show this property by induction on the size of M.

Since $\Gamma \vdash M \sim N' : \mathtt{HH} \to c'$, by the property stated in the beginning of this proof, we can distinguish four cases.

- If M is a nonce and $\Gamma(M) = \tau_M^{\mathtt{HH},a}$: then this contradicts Lemma 20. Indeed, this lemma (point 5) implies that $M \in \mathcal{BN}$, but also (by point 6), since $\Gamma \vdash M \sim N : \mathtt{LL} \to c$, that either $\Gamma(M) = \tau_M^{\mathtt{LL},a}$ for some $a \in \{1, \infty\}$, or $M \in \mathcal{FN} \cup \mathcal{C}$.

- If M is a key and $\Gamma(M,k) <: \text{key}^{\texttt{HH}}(T)$ for some T, k: then by Lemma 20, since $\Gamma \vdash M \sim N : \texttt{LL} \to c$, there exists T', k' such that $\Gamma(M,k') <: \text{key}^{\texttt{LL}}(T')$ or $k, k' \in \mathcal{FK}$. Since Γ is well-formed, and by Lemma 3, this contradicts $\Gamma(M) <: \text{key}^{\texttt{HH}}(T)$.
- If $\Gamma \vdash M \sim N' : \texttt{HH} * T \to c''$ for some T, c'': then by Lemma 19, since M, N' are ground, there exist M_1, M_2, N'_1, N'_2, c'_1 such that $M = \langle M_1, M_2\rangle$, $N' = \langle N'_1, N'_2\rangle$, and $\Gamma \vdash M_1 \sim N'_1 : \texttt{HH} \to c'_1$. Moreover, since $\Gamma \vdash M \sim N : \texttt{LL} \to c$, also by Lemma 19, there exist N_1, N_2, c_1 such that $N = \langle N_1, N_2\rangle$ and $\Gamma \vdash M_1 \sim N_1 : \texttt{LL} \to c_1$. However, by the induction hypothesis, $\Gamma \vdash M_1 \sim N'_1 : \texttt{HH} \to c'_1$ and $\Gamma \vdash M_1 \sim N_1 : \texttt{LL} \to c_1$ is impossible.
- If $\Gamma \vdash M \sim N' : T * \texttt{HH} \to c''$ for some T, c'': this case is similar to the previous one.

The next Lemma corresponds to Lemma 1.

Lemma 26 (Low terms are recipes on their constraints). *For all ground messages M, N, for all $T <: \texttt{LL}$, for all Γ, c, if $\Gamma \vdash M \sim N : T \to c$ then there exists an attacker recipe R without destructors such that $M = R(\phi_\ell(c) \cup \phi^\Gamma_{\texttt{LL}})$ and $N = R(\phi_r(c) \cup \phi^\Gamma_{\texttt{LL}})$.*

Proof. We prove this lemma by induction on the typing derivation of $\Gamma \vdash M \sim N : T \to c$. We distinguish several cases for the last rule in this derivation.

- TNONCE, THIGH, TOR, TLR^1, TLR$^\infty$, TLR', TLRVAR: these cases are not possible, since the type they give to terms is never a subtype of LL by Lemma 3.
- TVAR: this case is not possible since M, N are ground.
- TSUB: this case is directly proved by applying the induction hypothesis to the judgement $\Gamma \vdash M \sim N : T' \to c$ where $T' <: T <: \texttt{LL}$, which appears in the conditions of this rule, and has a shorter derivation.
- TLRL': in this case, $\Gamma \vdash M \sim N : [\![\tau_n^{\texttt{LL},a}\,;\, \tau_n^{\texttt{LL},a}]\!] \to c'$ for some nonce n, some $a \in \{\infty, 1\}$, some c', and $c = \emptyset$. By Lemma 16, this implies that $M = N = n$, and $\Gamma(n) = \tau_n^{\texttt{LL},a}$. Thus, by definition, there exists x such that $\phi^\Gamma_{\texttt{LL}}(x) = n$ and the claim holds with $R = x$.
- TNONCEL: in this case $M = N = n$ for some $n \in \mathcal{N}$ such that $\Gamma(n) = \tau_n^{\texttt{LL},a}$ for some $a \in \{1, \infty\}$. Hence, by definition, there exists x such that $\phi^\Gamma_{\texttt{LL}}(x) = n$ and the claim holds with $R = x$.
- TCSTFN: then $M = N = a \in \mathcal{C} \cup \mathcal{FN} \cup \mathcal{FK}$, and the claim holds with $R = a$.
- TKEY: then by well-formedness of Γ, $M = N = k \in \mathcal{K}$ and there exists T' such that $\Gamma(k,k) <: \text{key}^{\texttt{LL}}(T')$. By definition, there exists x such that $\phi^\Gamma_{\texttt{LL}}(x) = k$ and the claim holds with $R = x$.
- TPUBKEYL, TVKEYL: then $M = N = \texttt{pk}(k)$ (resp. $\texttt{vk}(k)$) for some $k \in \text{keys}(\Gamma) \cup \mathcal{FK}$. If $k \in \text{keys}(\Gamma)$, by definition, there exists x such that $\phi^\Gamma_{\texttt{LL}}(x) = \texttt{pk}(k)$ (resp. $\texttt{vk}(k)$) and the claim holds with $R = x$. If $k \in \mathcal{FK}$, the claim holds with $R = \texttt{pk}(k)$.
- TPUBKEY, TVKEY: these two cases are similar, we write the proof for the TPUBKEY case. The form of this rule application is:

$$\dfrac{\dfrac{\Pi}{\Gamma \vdash M \sim N : T \to \emptyset}}{\Gamma \vdash \texttt{pk}(M) \sim \texttt{pk}(N) : \text{pkey}(T) \to \emptyset}$$

  for some T such that $\text{pkey}(T) <: \texttt{LL}$. By Lemma 3, this implies that there exist T', l such that $T <: \text{eqkey}^l(T')$. Thus, $\Gamma \vdash M \sim N : \text{eqkey}^l(T') \to \emptyset$. By Lemma 20, this implies $M = N = k \in \text{keys}(\Gamma)$. By definition, there exists x such that $\phi^\Gamma_{\texttt{LL}}(x) = \texttt{pk}(k)$, and the claim holds with $R = x$.
- TPAIR, THASHL: these cases are similar. We detail the TPAIR case. In that case, $T = T_1 * T_2$ for some T_1, T_2. By Lemma 3, T_1, T_2 are subtypes of LL. In addition, there exist M_1, M_2, N_1, N_2, c_1, c_2 such that $\Gamma \vdash M_i \sim N_i : T_i \to c_i$ (for $i \in \{1,2\}$). By applying the induction hypothesis to these two judgements (which have shorter proofs), we obtain R_1, R_2 such that for all i, $M_i = R_i(\phi_\ell(c_i) \cup \phi^\Gamma_{\texttt{LL}})$ and $N_i = R_i(\phi_r(c_i) \cup \phi^\Gamma_{\texttt{LL}})$. Therefore the claim holds with $R = \langle R_1, R_2\rangle$.

- TEncH, TAencH, THash, TSignH: these four cases are similar. In each case, by the form of the typing rule, we have $c = \{M \sim N\} \cup c'$ for some c'. Therefore by definition of $\phi_\ell(c)$, $\phi_r(c)$, there exists x such that $\phi_\ell(c)(x) = M$ and $\phi_r(c)(N) = N$. The claim holds with $R = x$.
- TEncL, TAencL: these two cases are similar, we write the proof for the TEncL case. The form of this rule application is:

$$\dfrac{\dfrac{\Pi}{\Gamma \vdash M \sim N : (\mathtt{LL})_T \to c}}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}$$

  for some T such that $T = \mathtt{LL}$ or $T <: \mathrm{key}^{\mathtt{LL}}(T')$ for some T'. In both cases $T <: \mathtt{LL}$. By Lemma 17, there exist M', N', M'', N'', c', c'' such that $M = \mathtt{enc}(M', M'')$, $N = \mathtt{enc}(N', N'')$, $c = c' \cup c''$, $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'$ and $\Gamma \vdash M'' \sim N'' : T \to c''$, both with a proof shorter that Π. Thus by applying the induction hypothesis to these judgements, there exist R, R' such that $M' = R(\phi_\ell(c') \cup \phi^\Gamma_{\mathtt{LL}})$, $N' = R(\phi_r(c') \cup \phi^\Gamma_{\mathtt{LL}})$, $M'' = R'(\phi_\ell(c'') \cup \phi^\Gamma_{\mathtt{LL}})$, and $N'' = R'(\phi_r(c'') \cup \phi^\Gamma_{\mathtt{LL}})$.
  Therefore, the claim holds with the recipe $\mathtt{enc}(R, R')$.
- TEnc, TAenc: these two cases are similar, we write the proof for the TEnc case. The form of this rule application is:

$$\dfrac{\dfrac{\Pi}{\Gamma \vdash M \sim N : T \to c} \qquad \dfrac{\Pi'}{\Gamma \vdash M' \sim N' : T' \to c'}}{\Gamma \vdash \mathtt{enc}(M, M') \sim \mathtt{enc}(N, N') : (T)_{T'} \to c \cup c'}$$

  for some T, T' such that $(T)_{T'} <: \mathtt{LL}$. By Lemma 3, $T <: \mathtt{LL}$ and $T' <: \mathtt{LL}$. We conclude the proof of this case similarly to the TEncL case.
- TSignL: the form of this rule application is:

$$\dfrac{\dfrac{\Pi}{\Gamma \vdash M' \sim N' : \mathtt{LL} \to c} \qquad \dfrac{\Pi}{\Gamma \vdash M'' \sim N'' : \mathtt{LL} \to c'}}{\Gamma \vdash \mathtt{sign}(M', M'') \sim \mathtt{sign}(N', M'') : \mathtt{LL} \to c \cup c'}$$

  with $M = \mathtt{sign}(M', M'')$, $N = \mathtt{sign}(N', N'')$. Thus by applying the induction hypothesis to the two hypotheses of the rule, *i.e.* $\Gamma \vdash M' \sim N' : \mathtt{LL} \to c$ and $\Gamma \vdash M'' \sim N'' : \mathtt{LL} \to c'$ there exist R, R' such that $M' = R(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}})$, $N' = R(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$, $M'' = R'(\phi_\ell(c') \cup \phi^\Gamma_{\mathtt{LL}})$, and $N'' = R'(\phi_r(c') \cup \phi^\Gamma_{\mathtt{LL}})$.
  Therefore, the claim holds with the recipe $\mathtt{sign}(R, R')$.

Lemma 27 (Low frames with consistent constraints are statically equivalent). *For all ground ϕ, ϕ', for all c, Γ, if*

- $\Gamma \vdash \phi \sim \phi' : \mathtt{LL} \to c$
- *and c is consistent in $\Gamma_{\mathcal{N},\mathcal{K}}$,*

then ϕ and ϕ' are statically equivalent.

Proof. We can first notice that since ϕ and ϕ' are ground, so is c (this is easy to see by examining the typing rules for terms). Let R, R' be two attacker recipes, such that $\mathrm{vars}(R) \cup \mathrm{vars}(R') \subseteq \mathrm{dom}(\phi)(= \mathrm{dom}(\phi'))$.

For all $x \in \mathrm{dom}(\phi)(= \mathrm{dom}(\phi'))$, by assumption, there exists $c_x \subseteq c$ such that $\Gamma \vdash \phi(x) \sim \phi'(x) : \mathtt{LL} \to c_x$. By Lemma 26, there exists a recipe R_x such that $\phi(x) = R_x(\phi_\ell(c_x) \cup \phi^\Gamma_{\mathtt{LL}})$ and $\phi'(x) = R_x(\phi_r(c_x) \cup \phi^\Gamma_{\mathtt{LL}})$.

Since $c_x \subseteq c$, we also have $\phi(x) = R_x(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}})$ and $\phi'(x) = R_x(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$.

Let $\overline{R}$ and $\overline{R'}$ be the recipes obtained by replacing every occurence of x with R_x in respectively R and R', for all variable $x \in \mathrm{dom}(\phi)(= \mathrm{dom}(\phi'))$.

We then have $R\phi = \overline{R}(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}})$ and $R'\phi = \overline{R}'(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}})$; and similarly $R\phi' = \overline{R}(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$ and $R'\phi' = \overline{R}'(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$.

Since c is ground, and consistent in $\Gamma_{\mathcal{N},\mathcal{K}}$, by definition of consistency, the frames $\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}}$ and $\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}}$ are statically equivalent. Hence, by definition of static equivalence,

$$\overline{R}(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}}) = \overline{R}'(\phi_\ell(c) \cup \phi^\Gamma_{\mathtt{LL}}) \quad \iff \quad \overline{R}(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}}) = \overline{R}'(\phi_r(c) \cup \phi^\Gamma_{\mathtt{LL}})$$

i.e.

$$R\phi = R'\phi \quad \iff \quad R\phi' = R'\phi'$$

Therefore, ϕ and $.\phi'$ are statically equivalent.

We now prove the following invariant, which corresponds to Lemma 2.

Lemma 28 (Invariant). *For all Γ, ϕ_P, ϕ'_P, ϕ_Q, σ^1_P, σ^2_P, σ^1_Q,c_ϕ, c_σ, for all multisets of processes $\mathcal{P}$, $\mathcal{P}'$, $\mathcal{Q}$, where the processes in $\mathcal{P}$, $\mathcal{P}'$, $\mathcal{Q}$ are noted $\{P_i\}$, $\{P'_i\}$, $\{Q_i\}$; for all constraint sets $\{C_i\}$, if:*

- $|\mathcal{P}| = |\mathcal{Q}|$
- $\mathrm{dom}(\phi_P) = \mathrm{dom}(\phi_Q)$
- *$\forall i$, there is a derivation Π_i of $\Gamma \vdash P_i \sim Q_i \to C_i$,*
- $\Gamma \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$
- *for all $i \neq j$, the sets of bound variables in P_i and P_j (resp. Q_i and Q_j) are disjoint, and similarly for the names*
- *σ^1_P, σ^1_Q are ground, and there exist ground $\sigma_P \supseteq \sigma^1_P$, $\sigma_Q \supseteq \sigma^1_Q$ such that*
  - $(\mathrm{dom}(\sigma_P) \backslash \mathrm{dom}(\sigma^1_P)) \cap (\mathrm{vars}(\mathcal{P}) \cup \mathrm{vars}(\phi_P)) = \emptyset$,
  - $(\mathrm{dom}(\sigma_Q) \backslash \mathrm{dom}(\sigma^1_Q)) \cap (\mathrm{vars}(\mathcal{Q}) \cup \mathrm{vars}(\phi_Q)) = \emptyset$, *and*
  - $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$,
  - *for all $x \in \mathrm{dom}(\sigma_P)$, $\sigma_P(x) \downarrow = \sigma_P(x)$, and similarly for σ_Q,*
- $c_\sigma \subseteq [\![c_\phi]\!]_{\sigma^1_P,\sigma^1_Q}$,
- *$[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma^1_P,\sigma^1_Q}$ is consistent,*
- $(\mathcal{P}, \phi_P, \sigma^1_P) \xrightarrow{\alpha} (\mathcal{P}', \phi'_P, \sigma^2_P)$,

then there exist a word w, a multiset $\mathcal{Q}' = \{Q'_i\}$, constraint sets $\{C'_i\}$, a frame ϕ'_Q, a substitution σ'_Q, an environment Γ', constraints c'_ϕ, and c'_σ such that:

- $w =_\tau \alpha$
- $|\mathcal{P}'| = |\mathcal{Q}'|$
- *for all $i \neq j$, the sets of bound variables in P'_i and P'_j (resp. Q'_i and Q'_j) are disjoint, and similarly for the bound names;*
- $\mathrm{fvars}(\mathcal{P}') \cup \mathrm{fvars}(\phi'_P) \subseteq \mathrm{dom}(\sigma^2_P)$ *and* $\mathrm{fvars}(\mathcal{Q}') \cup \mathrm{fvars}(\phi'_Q) \subseteq \mathrm{dom}(\sigma^2_Q)$
- $\Gamma' \vdash \phi'_P \sim \phi'_Q : \mathtt{LL} \to c'_\phi$
- $(\mathcal{Q}, \phi_Q, \sigma^1_Q) \xrightarrow{w}_* (\mathcal{Q}', \phi'_Q, \sigma^2_Q)$,
- $\forall i, \Gamma' \vdash P'_i \sim Q'_i \to C'_i$,
- *σ^2_P, σ^2_Q are ground and there exist $\sigma'_P \supseteq \sigma^2_P$, and $\sigma'_Q \supseteq \sigma^2_Q$, such that*
  - $(\mathrm{dom}(\sigma'_P) \backslash \mathrm{dom}(\sigma^2_P)) \ \cap \ (\mathrm{vars}(\mathcal{P}') \cup \mathrm{vars}(\phi'_P)) = \emptyset$,
  - $(\mathrm{dom}(\sigma'_Q) \backslash \mathrm{dom}(\sigma^2_Q)) \ \cap \ (\mathrm{vars}(\mathcal{Q}') \cup \mathrm{vars}(\phi'_Q)) = \emptyset$, *and*
  - $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P \sim \sigma'_Q : \Gamma'_{\mathcal{X}} \to c'_\sigma$,
  - *for all $x \in \mathrm{dom}(\sigma'_P)$, $\sigma'_P(x) \downarrow = \sigma'_P(x)$, and similarly for σ'_Q,*
- $\mathrm{dom}(\phi'_P) = \mathrm{dom}(\phi'_Q)$,

- $c'_\sigma \subseteq \left[\!\left[c'_\phi \right]\!\right]_{\sigma_P^2,\sigma_Q^2}$,
- $\left[\!\left[(\cup_{\times i} C'_i) \cup_\forall c'_\phi \right]\!\right]_{\sigma_P^2,\sigma_Q^2}$ *is consistent.*

Proof. Note that the assumption that σ_P (resp. σ_Q) extends σ_P^1 (resp. σ_Q^1) only with variables not appearing in $\mathcal{P}$ or ϕ_P (resp. $\mathcal{Q}$ or ϕ_Q) implies that $[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P^1,\sigma_Q^1} = [\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q}$. Similarly, we have $\left[\!\left[(\cup_{\times i} C'_i) \cup_\forall c'_\phi \right]\!\right]_{\sigma_P^2,\sigma_Q^2} = \left[\!\left[(\cup_{\times i} C'_i) \cup_\forall c'_\phi \right]\!\right]_{\sigma'_P,\sigma'_Q}$, $[\![c_\phi]\!]_{\sigma_P^1,\sigma_Q^1} = [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$, and $\left[\!\left[c'_\phi \right]\!\right]_{\sigma_P^2,\sigma_Q^2} = \left[\!\left[c'_\phi \right]\!\right]_{\sigma'_P,\sigma'_Q}$.

First, we show that it is sufficient to prove this lemma in the case where Γ does not contain any union types. Indeed, assume we know the property holds in that case. Let us show that the lemma then also holds in the other case, *i.e.* if Γ contains union types. By hypothesis, σ_P, σ_Q are ground, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_\mathcal{X} \to c_\sigma$. Hence we know by Lemma 6 that there exists a branch $\Gamma'' \in \mathrm{branches}(\Gamma)$ (thus Γ'' does not contain union types), such that $(\Gamma'')_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : (\Gamma'')_\mathcal{X} \to c_\sigma$.

Moreover, by Lemma 9, $\forall i, \Gamma'' \vdash P_i \sim Q_i \to C''_i \subseteq C_i$; and by Lemma 7, $\Gamma'' \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$. In addition by Lemma 13, $[\![(\cup_{\times i} C''_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q}$ is a subset of $[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q}$ and is therefore consistent. Thus, if the lemma holds when the environment does not contain union types, it can be applied to the same processes, frames, substitutions and to Γ'', which directly concludes the proof.

Therefore, we may assume that Γ does not contain any union types.

Note that, since by assumption $c_\sigma \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$, we have

$$[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q} = [\![(\cup_{\times i} C''_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q} \cup_\forall c_\sigma.$$

Hence, by Lemma 13, $(\cup_{\times i} [\![C_i]\!]_{\sigma_P,\sigma_Q} \cup_\forall c_\sigma)$ is consistent. Thus the assumption on the disjointness of the sets of bound variables (and names) in the processes implies, using Lemma 10, that each of the $[\![C_i]\!]_{\sigma_P,\sigma_Q} \cup_\forall c_\sigma$ is also consistent. Moreover, this disjointness property for $\mathcal{P}'$ and $\mathcal{Q}'$ follows from the other points, as it is easily proved by examining the reduction rules that it is preserved by reduction.

By hypothesis, $(\mathcal{P}, \phi_P, \sigma_P^1)$ reduces to $(\mathcal{P}', \phi'_P, \sigma_P^2)$ We know from the form of the reduction rules that exactly one process $P_i \in \mathcal{P}$ is reduced, while the others are unchanged. By the assumptions, there is a corresponding process $Q_i \in \mathcal{Q}$ and a derivation Π_i of $\Gamma \vdash P_i \sim Q_i \to C_i$.

We continue the proof by a case disjunction on the last rule of Π_i. Let us first consider the cases of the rules PZERO, PPAR, PNEW, and POR.

- PZERO: then $P_i = Q_i = 0$. Hence, the reduction rule applied to $\mathcal{P}$ is Zero, and $\mathcal{P}' = \mathcal{P} \backslash \{P_i\}$, $\phi'_P = \phi_P$, and $\sigma_P^2 = \sigma_P^1$. The same reduction can be performed in $\mathcal{Q}$:

  $$(\mathcal{Q}, \phi_Q, \sigma_Q^1) \xrightarrow{\tau} (\mathcal{Q} \backslash \{Q_i\}, \phi_Q, \sigma_Q^1)$$

  Since the other processes, the frames, environments and substitutions do not change in this reduction, all the claims clearly hold in this case (with $\sigma'_P = \sigma_P$, $\sigma'_Q = \sigma_Q$, $c'_\phi = c_\phi$, $c'_\sigma = c_\sigma$). In particular, the consistency of the constraints follows from the consistency hypothesis. Indeed,

  $$\begin{aligned}(\cup_{\times j \neq i} C_j) \cup_\times C_i \cup_\forall c_\phi &= (\cup_{\times j \neq i} C_j) \cup_\times \{(\emptyset, \Gamma)\} \cup_\forall c_\phi \\ &= (\cup_{\times j \neq i} C_j) \cup_\forall c_\phi,\end{aligned}$$

  since Γ is already contained in the environments appearing in each C_j (by Lemma 14). Thus

  $$\left[\!\left[(\cup_{\times j} C'_j) \cup_\forall c'_\phi \right]\!\right]_{\sigma'_P,\sigma'_Q} = \left[\!\left[(\cup_{\times j} C_j) \cup_\forall c_\phi \right]\!\right]_{\sigma_P,\sigma_Q}$$

– PPAR: then $P_i = P_i^1 \mid P_i^2$, $Q_i = Q_i^1 \mid Q_i^2$. Hence, the reduction rule applied to $\mathcal{P}$ is Par:

$$(\mathcal{P}, \phi_P, \sigma_P^1) \xrightarrow{\tau} (\mathcal{P}\backslash\{P_i\} \cup \{P_i^1, P_i^2\}, \phi_P, \sigma_P^1).$$

We choose $\Gamma' = \Gamma$.
In addition

$$\Pi_i = \frac{\dfrac{\Pi^1}{\Gamma \vdash P_i^1 \sim Q_i^1 \to C_i^1} \qquad \dfrac{\Pi^2}{\Gamma \vdash P_i^2 \sim Q_i^2 \to C_i^2}}{\Gamma \vdash P_i \sim Q_i \to C_i = C_i^1 \cup_\times C_i^2}.$$

The same reduction rule can be applied to $\mathcal{Q}$:

$$(\mathcal{Q}, \phi_Q, \sigma_Q^1) \xrightarrow{\tau} (\mathcal{Q}\backslash\{Q_i\} \cup \{Q_i^1, Q_i^2\}, \phi_Q, \sigma_Q^1)$$

In this case again, the claims on the substitutions and frames, as well as the claim that $c'_\sigma \subseteq \left[\!\left[c'_\phi \right]\!\right]_{\sigma_P^2, \sigma_Q^2}$, hold since they do not change in the reduction. Moreover the processes in $\mathcal{P}'$ and $\mathcal{Q}'$ are still pairwise typably equivalent. Indeed, all the processes from $\mathcal{P}$ and $\mathcal{Q}$ are unchanged, except for P_i and Q_i which are reduced to P_i^1, P_i^2, Q_i^1, Q_i^2, and those are typably equivalent using Π^1 and Π^2.
Finally the constraint set is still consistent, since:

$$\begin{aligned}\left[\!\left[(\cup_{\times j} C'_j) \cup_\forall c'_\phi \right]\!\right]_{\sigma'_P, \sigma'_Q} &= \left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i^1 \cup_\times C_i^2 \cup_\forall c_\phi \right]\!\right]_{\sigma_P, \sigma_Q} \\ &= \left[\!\left[(\cup_{\times j} C_j) \cup_\forall c_\phi \right]\!\right]_{\sigma_P, \sigma_Q}\end{aligned}$$

– PNEW: then $P_i = \texttt{new}\ n : \tau_n^{l,a}.P'_i$ and $Q_i = \texttt{new}\ n : \tau_n^{l,a}.Q'_i$. P_i is reduced to P'_i by rule New:

$$(\mathcal{P}, \phi_P, \sigma_P^1) \xrightarrow{\tau} (\mathcal{P}\backslash\{P_i\} \cup \{P'_i\}, \phi_P, \sigma_P^1).$$

In addition

$$\Pi_i = \frac{\dfrac{\Pi'_i}{\Gamma, n : \tau_n^{l,a} \vdash P'_i \sim Q'_i \to C_i}}{\Gamma \vdash P_i \sim Q_i \to C_i}.$$

We choose $\Gamma' = \Gamma, n : \tau_n^{l,a}$.
The same reduction rule can be applied to $\mathcal{Q}$:

$$(\mathcal{Q}, \phi_Q, \sigma_Q^1) \xrightarrow{\tau} (\mathcal{Q}\backslash\{Q_i\} \cup \{Q'_i\}, \phi_Q, \sigma_Q^1)$$

The claim clearly holds. Indeed the processes are still pairwise typable:

- using Π'_i in the case of P'_i and Q'_i;
- using Π_j, for $j \neq i$, as well as Lemma 12, for the other processes, since n does not appear in these processes by assumption.

In addition, all the frames, substitutions, and constraints are unchanged; and σ, σ' are well-typed in Γ' if and only if they are well-typed in Γ.

– PNEWKEY: This case is analogous to the PNEW case.
– POr: this case is not possible, since we have already eliminated the case where Γ contains union types.

In all the other cases for the last rule in Π_i, we know that the head symbol of P_i is not $|$, $\mathtt{0}$ or $\mathtt{new}$.

Hence, the form of the reduction rules implies that $P_i \in \mathcal{P}$ is reduced to exactly one process $P'_i \in \mathcal{P}'$, while the other processes in $\mathcal{P}$ do not change (*i.e.* $P'_j = P_j$ for $j \neq i$). If we show in each case that the same reduction rule that is applied to P_i can be applied to reduce $\mathcal{Q}$ to a multiset $\mathcal{Q}'$ by reducing process Q_i into Q'_i, we will also have $Q'_j = Q_j$ for $j \neq i$. Therefore the claim on the cardinality of the processes multisets will hold.

Since P_i, Q_i can be typed and the head symbol of P_i is not $\mathtt{new}$, it is clear by examining the typing rules that the head symbol of Q_i is not $\mathtt{new}$ either. Hence, we will choose a Γ' containing the same nonces and keys as Γ.

The proofs for theses cases follow the same structure:

- The typing rule gives us information on the form of P_i and Q_i.
- The form of P_i gives us information on which reduction rule was applied to $\mathcal{P}$.
- The form of Q_i is the same as P_i. Hence (additional conditions may need to be checked depending on the rule) Q_i can be reduced to some process Q'_i by applying the same reduction rule that was applied to P_i (or at least, a reduction rule with the same action).
- thus $\mathcal{Q}$ can be reduced too, with the same actions as $\mathcal{P}$. We then check the additional conditions on the typing of the processes, frames and substitutions, and the consistency condition.

First, let us consider the POUT case.

- POUT: then $P_i = \mathtt{out}(M).P'_i$ and reduces to P'_i via the Out rule, and $Q_i = \mathtt{out}(N).Q'_i$ for some N and Q'_i. Since the Out rule can be applied to P_i, $M\sigma^1_P \downarrow\neq \bot$, *i.e.* $M\sigma^1_P \downarrow= M\sigma^1_P$. In addition

$$\Pi_i = \frac{\Pi \qquad \Pi'}{\dfrac{\Gamma \vdash P'_i \sim Q'_i \to C'_i \qquad \Gamma \vdash M \sim N : \mathtt{LL} \to c}{\Gamma \vdash P_i \sim Q_i \to C_i = C'_i \cup_\forall c}}.$$

We have $\sigma^2_P = \sigma^1_P$, $\phi'_P = \phi_P \cup \{M/ax_n\}$, and $\alpha = \mathtt{new}\ ax_n.\mathtt{out}(ax_n)$.
Since $\Gamma \vdash M \sim N : \mathtt{LL} \to c$ and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$, by Lemma 24, we have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma_P \sim N\sigma_Q : \mathtt{LL} \to c'$ for some c'. That is to say $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma^1_P \sim N\sigma^1_Q : \mathtt{LL} \to c'$. Since we also know that $M\sigma^1_P \downarrow\neq \bot$, by Lemma 22, we also have $N\sigma^1_Q \downarrow\neq \bot$.
Hence, the same reduction rule Out can be applied to reduce the process Q_i into Q'_i, and the claim on the reduction of $\mathcal{Q}$ holds. We choose $\Gamma' = \Gamma$. We have $\sigma^2_Q = \sigma^1_Q$, and $\phi'_Q = \phi_Q \cup \{N/ax_n\}$. We also choose $\sigma'_P = \sigma_P$, $\sigma'_Q = \sigma_Q$, $c'_\phi = c_\phi \cup c$ and $c'_\sigma = c_\sigma$. The substitutions σ^1_P, σ^1_Q are not extended by the reduction, and the typing environment does not change, which trivially proves the claim regarding the substitutions.
In addition, since by assumption $c_\sigma \subseteq [\![c_\phi]\!]_{\sigma^1_P,\sigma^1_Q}$, and since $c_\phi \subseteq c'_\phi$, we have $c'_\sigma \subseteq \left[\!\left[c'_\phi\right]\!\right]_{\sigma^2_P,\sigma^2_Q}$.
Moreover, since only M and N are added to the frames in the reduction, Π' suffices to prove the claim that $\Gamma \vdash \phi'_P \sim \phi'_Q : \mathtt{LL} \to c'_\phi$. Since all processes other that P_i and Q_i are unchanged by the reduction (and since the typing environment is also unchanged), Π suffices to proves the claim that $\forall j.\ \Gamma' \vdash P'_j \sim Q'_j \to C'_j$ (with $C'_j = C_j$ for $j \neq i$).
Thus, in this case, it only remains to be proved that $\left[\!\left[(\cup_{\times j} C'_j) \cup_\forall c'_\phi\right]\!\right]_{\sigma'_P,\sigma'_Q}$ is consistent. This constraint set is equal to

$$\left[\!\left[(\cup_{\times j\neq i} C_j) \cup_\times C'_i \cup_\forall (c_\phi \cup c)\right]\!\right]_{\sigma_P,\sigma_Q}$$

i.e. to

$$\left[\!\left[(\cup_{\times j\neq i} C_j) \cup_\times (C'_i \cup_\forall c) \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

i.e.

$$\left[\!\left[(\cup_{\times j} C_j)\cup_{\forall} c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

which is consistent by hypothesis. Hence the claim holds in this case.

In the remaining cases, from the form of the typing rules for processes, the head symbol of neither P_i nor Q_i is $\mathtt{out}$. Thus, the reduction applied to P_i (from the assumption), as well as the one applied to Q_i (which, as we will show, has the same action as the rule for P_i), cannot be Out. Therefore no new term is output on either side, and $\phi'_P = \phi_P$ and $\phi'_Q = \phi_Q$. Hence the claim on the domains of the frames holds by assumption. Moreover, as we will see, in all cases Γ' is either Γ, or $\Gamma, x : T$ where x is a variable bound in (the head of) P_i and Q_i, and T is not a union type.

We choose $c'_\phi = c_\phi$. The claim that $\Gamma' \vdash \phi'_P \sim \phi'_Q : \mathtt{LL} \to c'_\phi$ is then actually that $\Gamma' \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$, which is true by Lemma 12, since by hypothesis $\Gamma \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$.

Besides, in the cases where we choose $\Gamma' = \Gamma$ then it is true (by hypothesis) that for $j \neq i$, $\Gamma' \vdash P'_j \sim Q'_j \to C_j$. In the cases where we choose $\Gamma' = \Gamma, x : T$, where x is bound in P_i and Q_i, then, since the processes are assumed to use different variable names, x does not appear in P_j or Q_j (for $j \neq i$). Hence, if $j \neq i$, using the assumption that $\Gamma \vdash P_j \sim Q_j \to C_j$, by Lemma 12, we have $\Gamma' \vdash P'_j \sim Q'_j \to C'_j$, where $C'_j = \{(c, \Gamma_c \cup \{x : T\}) | (c, \Gamma_c) \in C_j\}$.

Hence, for each remaining possible last rule of Π_i, we only have to show that:

a) The same reduction rule can be applied to Q_i as to P_i, with the same action. (Except in the case of the rule PIFLR, as we will see, where rule If-Then may be applied on one side while rule If-Else is applied on the other side, but this has no influence on the argument, as these two rules both represent a silent action, and have a very similar form.)

b) there exist σ'_P and σ'_Q ground, and containing σ^2_P and σ^2_Q respectively, that satisfy the conditions on the domains, contain only messages that do not reduce to $\bot$, and such that $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P \sim \sigma'_Q : \Gamma'_{\mathcal{X}} \to c'_\sigma$ for some set of constraints c'_σ. Since at most one variable x is added to the substitutions in the reduction, we will show in each case that we can choose these substitutions such that either $\sigma'_P = \sigma_P$ and $\sigma'_Q = \sigma_Q$; or $\sigma'_P = \sigma_P \cup \{M/x\}$ and $\sigma'_Q = \sigma_Q \cup \{N/x\}$ for some messages M, N. In all cases, it is clear from the reduction rules that $M \downarrow \neq \bot$ and $N \downarrow \neq \bot$. We will then only need to check the well-typedness condition on variable x, *i.e.* $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P(x) \sim \sigma'_Q(x) : \Gamma'(x) \to c_x$ for some c_x. We can then choose $c'_\sigma = c_\sigma \cup c_x$. As we will see in the proof, we will always have $c_x \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$.

In addition, $c'_\phi = c_\phi$, and by assumption, x cannot appear in c_ϕ, thus $\left[\!\left[c'_\phi\right]\!\right]_{\sigma'_P,\sigma'_Q} = [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$. Therefore, since by assumption $c_\sigma \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$, the claim that $c'_\sigma \subseteq \left[\!\left[c'_\phi\right]\!\right]_{\sigma'_P,\sigma'_Q}$ will always hold.

c) the new processes obtained by reducing P_i and Q_i are typably equivalent in Γ', with a constraint C'_i, such that

$$\left[\!\left[(\cup_{\times j\neq i} C_j)\cup_{\times} C'_i \cup_{\forall} c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

is consistent.

The actual claim, from the statement of the lemma, is that

$$\left[\!\left[(\cup_{\times j\neq i} C'_j)\cup_{\times} C'_i \cup_{\forall} c_\phi\right]\!\right]_{\sigma'_P,\sigma'_Q}$$

is consistent, but we can show that the previous condition is sufficient.

In the case where $\Gamma = \Gamma'$, we have $\sigma'_P = \sigma_P$, $\sigma'_Q = \sigma_Q$, $C'_j = C_j$ for $j \neq i$, and $c'_\sigma = c_\sigma$. Thus the proposed condition is clearly sufficient (it is even necessary in this case).
In the case where $\Gamma' = \Gamma, x : T$ for some T which is not a union type, and the substitutions σ'_P, σ'_Q are σ_P, σ_Q extended with a term associated to x, the proof that the condition is sufficient is more involved. First, we show that $(\cup_{\times\, j\neq i} C'_j)\cup_\times C'_i = (\cup_{\times\, j\neq i} C_j)\cup_\times C'_i$. Indeed, if S denotes the set $(\cup_{\times\, j\neq i} C'_j)\cup_\times C'_i$, we have

$$\begin{aligned}
S &= \{(\bigcup_j c'_j, \bigcup_j \Gamma'_j) \mid \forall j.\ (c'_j, \Gamma'_j) \in C'_j \ \wedge\ \forall j, j'.\ \Gamma'_j \text{ and } \Gamma'_{j'} \text{ are compatible})\} \\
&= \{(c'_i \cup \bigcup_{j\neq i} c_j, \Gamma'_i \cup \bigcup_{j\neq i} (\Gamma_j, x : T) \mid (c'_i, \Gamma'_i) \in C'_i \ \wedge\ (\forall j \neq i.\ (c_j, \Gamma_j) \in C_j) \wedge \\
&\qquad (\forall j \neq i, j' \neq i.\ (\Gamma_j, x : T) \text{ and } (\Gamma_{j'}, x : T) \text{ are compatible}) \wedge \\
&\qquad (\forall j \neq i.\ \Gamma'_i \text{ and } (\Gamma_j, x : T) \text{ are compatible}))\}
\end{aligned}$$

since we already know that for $j \neq i$, $C'_j = \{(c, \Gamma_c \cup \{x : T\}) | (c, \Gamma_c) \in C_j\}$. Assuming we show that $\Gamma, x : T \vdash P'_i \sim Q'_i \to C'_i$, by Lemma 14, we will also have that all the Γ'_i appearing in the elements of C'_i contain $x : T$ (since T is not a union type). Hence:

$$\begin{aligned}
S &= \{(c'_i \cup \bigcup_{j\neq i} c_j, \Gamma'_i \cup (\bigcup_{j\neq i} \Gamma_j) \mid (c'_i, \Gamma'_i) \in C'_i \ \wedge\ (\forall j \neq i.\ (c_j, \Gamma_j) \in C_j) \wedge \\
&\qquad (\forall j \neq i, j' \neq i.\ \Gamma_j \text{ and } \Gamma_{j'} \text{ are compatible}) \ \wedge\ (\forall j \neq i.\ \Gamma'_i \text{ and } \Gamma_j \text{ are compatible}))\} \\
&= (\cup_{\times\, j\neq i} C_j)\cup_\times C'_i
\end{aligned}$$

It is thus sufficient to ensure the consistency of $[\![S \cup_\forall c_\phi]\!]_{\sigma'_P, \sigma'_Q}$. Since $\sigma'_P = \sigma_P \cup \{\sigma'_P(x)/x\}$ (and similarly for Q), it then suffices to show the consistency of $\left[\!\left[[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q} \right]\!\right]_{\sigma'_P(x)/x, \sigma'_Q(x)/x}$.
The substitutions $\sigma'_P(x)$ and $\sigma'_Q(x)$ are ground, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P(x) \sim \sigma'_Q(x) : T \to c_x$ (which we will show for each case as point b)). Hence by Lemma 13, if $[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q} \cup_\forall c_x$ is consistent, then $\left[\!\left[[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q} \right]\!\right]_{\sigma'_P(x)/x, \sigma'_Q(x)/x}$ is consistent. Moreover, as explained in point b), we will show in each case that $c_x \subseteq [\![c_\phi]\!]_{\sigma_P, \sigma_Q}$. Thus $[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q} \cup_\forall c_x = [\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q}$.
Therefore, by the previous implication, it is sufficient to prove that $[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q}$ is consistent, to ensure that $\left[\!\left[[\![S \cup_\forall c_\phi]\!]_{\sigma_P, \sigma_Q} \right]\!\right]_{\sigma'_P(x)/x, \sigma'_Q(x)/x}$ is consistent. This is the condition stated at the beginning of this point, since $S = (\cup_{\times\, j\neq i} C_j)\cup_\times C'_i$.

We can now prove the remaining cases for the last rule of Π_i, that is to say the cases of the rules PIN, PLET, PLETDEC, PLETADECSAME, PLETADECDIFF, PLETLRK, PIFL, PIFLR, PIFS, PIFLR*, PIFP, PIFI, PIFLR'*, and PIFALL.

– PIN: then $P_i = \mathtt{in}(x).P'_i$ and reduces to P'_i via the In rule, and $Q_i = \mathtt{in}(x).Q'_i$ for some Q'_i. In addition

$$\Pi_i = \dfrac{\dfrac{\Pi}{\Gamma, x : \mathtt{LL} \vdash P'_i \sim Q'_i \to C'_i}}{\Gamma \vdash P_i \sim Q_i \to C_i = C'_i}.$$

We have $\alpha = \mathtt{in}(R)$ for some attacker recipe R such that $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi_P)$, and $R\phi_P\sigma_P^1 \downarrow = R\phi_P\sigma_P \downarrow \neq \bot$. We also have $\sigma_P^2 = \sigma_P^1 \cup \{R\phi_P\sigma_P^1 \downarrow /x\}$, $\phi'_P = \phi_P$.
The same reduction rule In can be applied to reduce the process Q_i into Q'. Indeed,

- $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi_Q)$ since $\mathrm{dom}(\phi_Q) = \mathrm{dom}(\phi_P)$ by hypothesis;
- $R\phi_Q\sigma_Q^1 \downarrow = R\phi_Q\sigma_Q \downarrow \neq \bot$. This follows from Lemma 23, using the fact that by Lemma 24, $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \phi_P\sigma_P \sim \phi_Q\sigma_Q : \mathtt{LL} \to c$, for some $c \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$.

Therefore point a) holds.
We choose $\Gamma' = \Gamma, x : \mathtt{LL}$. We have $\sigma_Q^2 = \sigma_Q^1 \cup \{R\phi_Q\sigma_Q^1 \downarrow /x\}$. We choose $\sigma'_P = \sigma_P \cup \{R\phi_P\sigma_P^1 \downarrow /x\}$ and $\sigma'_Q = \sigma_Q \cup \{R\phi_Q\sigma_Q^1 \downarrow /x\}$.
Lemmas 24 and 23, previously evoked, guarantee that

$$\Gamma_{\mathcal{N},\mathcal{K}} \vdash R\phi_P\sigma_P \downarrow \sim R\phi_Q\sigma_Q \downarrow : \mathtt{LL} \to c'$$

for some $c' \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$. This proves point b).
Moreover, Π and the fact that

$$\left[\!\!\left[(\cup_{\times\, j\neq i} C_j) \cup_\times C'_i \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q} = \left[\!\!\left[(\cup_{\times\, j} C_j) \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q}$$

which is consistent by hypothesis, prove point c) and conclude this case.

- PLET: then $P_i = \mathtt{let}\ x = t\ \mathtt{in}\ P'_i\ \mathtt{else}\ P''_i$ and $Q_i = \mathtt{let}\ x = t'\ \mathtt{in}\ Q'_i\ \mathtt{else}\ Q''_i$ for some P'_i, P''_i, Q'_i, Q''_i, t, t'. P_i reduces to either P'_i via the Let-In rule, or P''_i via the Let-Else rule. In addition

$$\Pi_i = \frac{x \notin \mathrm{dom}(\Gamma) \qquad \dfrac{\Pi}{\Gamma \vdash_d t \sim t' : T} \qquad \dfrac{\Pi'}{\Gamma, x : T \vdash P'_i \sim Q'_i \to C'_i} \qquad \dfrac{\Pi''}{\Gamma \vdash P''_i \sim Q''_i \to C''_i}}{\Gamma \vdash P_i \sim Q_i \to C_i = C'_i \cup C''_i}.$$

We have $\alpha = \tau$.
Let $\sigma = \sigma_P|_{\mathrm{vars}(t)\cup\mathrm{vars}(t')}$, $\sigma' = \sigma_Q|_{\mathrm{vars}(t)\cup\mathrm{vars}(t')}$, and $\Gamma'' = \Gamma_{\mathcal{N},\mathcal{K}} \cup (\Gamma|_{\mathrm{vars}(t)\cup\mathrm{vars}(t')})$. Since by assumption $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_\mathcal{X} \to c_\sigma$, we have $\Gamma''_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma''_\mathcal{X} \to c''$ for some $c'' \subset c_\sigma$. Hence, by Lemma 21, using Π, we have:

$$t\sigma \downarrow \neq \bot \iff t'\sigma' \downarrow \neq \bot$$

i.e.

$$t\sigma_P^1 \downarrow \neq \bot \iff t'\sigma_Q^1 \downarrow \neq \bot$$

Therefore, if rule Let-In is applied to P_i then it can also be applied to reduce Q_i into Q'_i, and if the rule applied to P_i is Let-Else then it can also be applied to reduce Q_i into Q''_i. This proves point a). We prove here the Let-In case. The Let-Else case is similar (although slightly easier, since no new variable is added to the substitutions).
In this case we have $\sigma_P^2 = \sigma_P^1 \cup \{t\sigma_P^1 \downarrow /x\}$ and $\sigma_Q^2 = \sigma_Q^1 \cup \{t'\sigma_Q^1 \downarrow /x\}$.
In addition, by hypothesis, $t\sigma_P = t\sigma_P^1 = t\sigma$ and $t'\sigma_Q = t'\sigma_Q^1 = t'\sigma'$.
By Lemma 21, we know in this case that there exists $c \subseteq c''$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash t\sigma_P \downarrow \sim t'\sigma_Q \downarrow : T \to c$.
Thus, by Lemma 4, there exists $T' \in \mathrm{branches}(T)$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash t\sigma_P \downarrow \sim t'\sigma_Q \downarrow : T' \to c$.
We choose $\Gamma' = \Gamma, x : T'$, $\sigma'_P = \sigma_P \cup \{t\sigma_P \downarrow /x\}$ and $\sigma'_Q = \sigma_Q \cup \{t'\sigma_Q \downarrow /x\}$. Since Γ does not contain union types, $\Gamma' \in \mathrm{branches}(\Gamma, x : T)$.
Since $c \subseteq c'' \subseteq c_\sigma$ and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash t\sigma_P \downarrow \sim t'\sigma_Q \downarrow : T' \to c$, point b) holds.
We now prove that point c) holds. Using Π', we have $\Gamma, x : T \vdash P'_i \sim Q'_i \to C'_i$. Hence, by Lemma 9, there exists $C'''_i \subseteq C'_i (\subseteq C_i)$ such that $\Gamma' \vdash P'_i \sim Q'_i \to C'''_i$.

Since $C_i''' \subseteq C_i$, we have

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'''\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}.$$

This last constraint set is consistent by hypothesis.
Hence, by Lemma 13, $\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'''\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is also consistent. This proves point c) and concludes this case.

– PLETDEC: then there exist y, P_i', P_i'', Q_i', Q_i'' such that $P_i = \texttt{let}\ x = \texttt{dec}(y, k_1)\ \texttt{in}\ P_i'\ \texttt{else}\ P_i''$, and $Q_i = \texttt{let}\ x = \texttt{dec}(y, k_2)\ \texttt{in}\ Q_i'\ \texttt{else}\ Q_i''$, and $\Gamma(y) = \texttt{LL}$. P_i reduces to either P_i' via the Let-In rule, or P_i'' via the Let-Else rule. In addition

$$\Pi_i = \frac{\Gamma(y) = \texttt{LL} \qquad \Gamma(k_1, k_2) = \text{key}^{\texttt{HH}}(T) \qquad \dfrac{\Pi_i'}{\Gamma, x : T \vdash P_i' \sim Q_i' \to C_i'} \qquad \dfrac{\Pi_i''}{\Gamma \vdash P_i'' \sim Q_i'' \to C_i''}}{\begin{array}{c}(\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_1, k_3) = \text{key}^{\texttt{HH}}(T') \Rightarrow \dfrac{\Pi_i^{1,k_3}}{\Gamma, x : T' \vdash P_i' \sim Q_i'' \to C_{k_3}}) \\ (\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_3, k_2) = \text{key}^{\texttt{HH}}(T') \Rightarrow \dfrac{\Pi_i^{2,k_3}}{\Gamma, x : T' \vdash P_i'' \sim Q_i' \to C'_{k_3}}) \\ \hline \Gamma \vdash \texttt{let}\ x = \texttt{dec}(y, k_1)\ \texttt{in}\ P_i'\ \texttt{else}\ P_i'' \sim \texttt{let}\ x = \texttt{dec}(y, k_2)\ \texttt{in}\ Q_i'\ \texttt{else}\ Q_i'' \to \\ C_i' \cup C_i'' \cup (\bigcup_{k_3} C_{k_3}) \cup (\bigcup_{k_3} {C'}_{k_3})\end{array}}$$

We have $\alpha = \tau$. In addition, by hypothesis, $\sigma_P(y) = \sigma_P^1(y)$ and $\sigma_Q(y) = \sigma_Q^1(y)$.
We consider two cases.

- If $\texttt{dec}(y\sigma_P, k_1) \downarrow \neq \bot$ then the reduction applied to P_i is Let-In, and P_i is reduced to P_i'. This also implies that there exists M such that $y\sigma_P = \texttt{enc}(M, k_1)$. Since $\Gamma(y) = \texttt{LL}$, we know by assumption that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash y\sigma_P \sim y\sigma_Q : \texttt{LL} \to c$ for some constraint $c \subseteq c_\sigma$. Hence, by Lemma 17, there exist N, k, T', $c' \subseteq c_\sigma$ such that $y\sigma_Q = \texttt{enc}(N, k)$, $\Gamma(k_1, k) = \text{key}^{\texttt{HH}}(T)$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T' \to c'$. Thus, by Lemma 4, there exists $T'' \in \text{branches}(T')$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T'' \to c'$.
  Two cases are possible.
  * If $k = k_2$, then $\texttt{dec}(y\sigma_Q, k_2) \downarrow = N$, and rule Let-In can be applied to reduce Q_i into Q_i', which proves point a). In this case we have $\sigma_P^2 = \sigma_P^1 \cup \{M/x\}$ and $\sigma_Q^2 = \sigma_Q^1 \cup \{N/x\}$.
    We choose $\Gamma' = \Gamma, x : T'$, $\sigma_P' = \sigma_P \cup \{M/x\}$ and $\sigma_Q' = \sigma_Q \cup \{N/x\}$ Since Γ does not contain union types, $\Gamma' \in \text{branches}(\Gamma, x : T)$.
    Since $c' \subseteq c_\sigma$ and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T'' \to c'$, point b) holds.
    We now prove that point c) holds. Using Π_i', we have $\Gamma, x : T \vdash P_i' \sim Q_i' \to C_i'$. Hence, by Lemma 9, there exists $C_i''' \subseteq C_i' (\subseteq C_i)$ such that $\Gamma' \vdash P_i' \sim Q_i' \to C_i'''$.
    Since $C_i''' \subseteq C_i$, we have

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'''\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}.$$

    This last constraint set is consistent by hypothesis.
    Hence, by Lemma 13, $\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'''\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is also consistent. This proves point c) and concludes this case.

* If $k \neq k_2$, then $\mathtt{dec}(y\sigma_Q, k_2) \downarrow = \bot$, and rule Let-Else can be applied to reduce Q_i into Q_i'', which proves point a). In this case we have $\sigma_P^2 = \sigma_P^1 \cup \{M/x\}$ and $\sigma_Q'^2 = \sigma_Q^1$.
We choose $\Gamma' = \Gamma, x : T'$, $\sigma'_P = \sigma_P \cup \{M/x\}$ and $\sigma'_Q = \sigma_Q \cup \{N/x\}$ (by well-formedness of the processes, x does not appear in Q_i''). Since Γ does not contain union types, $\Gamma' \in \mathrm{branches}(\Gamma, x : T)$.
Since $c' \subseteq c_\sigma$ and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T'' \to c'$, point b) holds.
We now prove that point c) holds. Using $\Pi_i^{1,k}$, we have $\Gamma, x : T \vdash P_i' \sim Q_i'' \to C_k$. Hence, by Lemma 9, there exists $C_i''' \subseteq C_k (\subseteq C_i)$ such that $\Gamma' \vdash P_i' \sim Q_i'' \to C_i'''$.
Since $C_i''' \subseteq C_i$, we have

$$\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i''' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times j} C_j) \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}.$$

This last constraint set is consistent by hypothesis.
Hence, by Lemma 13, $\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i''' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is also consistent. This proves point c) and concludes this case.

- If $\mathtt{dec}(y\sigma_P, k_1) \downarrow = \bot$ then the reduction applied to P_i is Let-Else, and P_i is reduced to P_i''. Again we distinguish two cases.
  * If $\mathtt{dec}(y\sigma_Q, k_2) \downarrow \neq \bot$ then rule Let-In can be applied to reduce Q_i into Q_i'. This also implies that there exists N such that $y\sigma_Q = \mathtt{enc}(N, k_2)$. Since $\Gamma(y) = \mathtt{LL}$, we know by assumption that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash y\sigma_P \sim y\sigma_Q : \mathtt{LL} \to c$ for some constraint $c \subseteq c_\sigma$. Hence, by Lemma 17, there exist M, k, T', $c' \subseteq c_\sigma$ such that $y\sigma_P = \mathtt{enc}(M, k)$, $\Gamma(k, k_2) = \mathrm{key}^{\mathtt{HH}}(T)$, and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T' \to c'$. Thus, by Lemma 4, there exists $T'' \in \mathrm{branches}(T')$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T'' \to c'$.
    In this case we have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q'^2 = \sigma_Q^1 \cup \{N/x\}$. We choose $\Gamma' = \Gamma, x : T'$, $\sigma'_P = \sigma_P \cup \{M/x\}$ and $\sigma'_Q = \sigma_Q \cup \{N/x\}$ (by well-formedness of the processes, x does not appear in P_i''). Since Γ does not contain union types, $\Gamma' \in \mathrm{branches}(\Gamma, x : T)$.
    Since $c' \subseteq c_\sigma$ and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M \sim N : T'' \to c'$, point b) holds.
    We now prove that point c) holds. Using $\Pi_i^{2,k}$, we have $\Gamma, x : T \vdash P_i'' \sim Q_i' \to C_k'$. Hence, by Lemma 9, there exists $C_i''' \subseteq C_k' (\subseteq C_i)$ such that $\Gamma' \vdash P_i'' \sim Q_i' \to C_i'''$.
    Since $C_i''' \subseteq C_i$, we have

$$\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i''' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times j} C_j) \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}.$$

    This last constraint set is consistent by hypothesis.
    Hence, by Lemma 13, $\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i''' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is also consistent. This proves point c) and concludes this case.
  * If $\mathtt{dec}(y\sigma_Q, k_2) \downarrow = \bot$ then rule Let-Else can be applied to reduce Q_i into Q_i''. In this case we have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q'^2 = \sigma_Q^1$. We choose $\Gamma' = \Gamma$, $\sigma'_P = \sigma_P$ and $\sigma'_Q = \sigma_Q$. Since the substitutions and environments do not change, point b) clearly holds.
    We now prove that point c) holds. Using Π_i'', we have $\Gamma \vdash P_i'' \sim Q_i'' \to C_i''$.
    Since $C_i'' \subseteq C_i$, we have

$$\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i'' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times j} C_j) \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}.$$

    This last constraint set is consistent by hypothesis.
    Hence, by Lemma 13, $\left[\!\left[(\cup_{\times j \neq i} C_j) \cup_\times C_i'' \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is also consistent. This proves point c) and concludes this case.

– PLETADECSAME and PLETADECDIFF: these two cases are similar to the PLETDEC case.
– PLETLRK: then $P_i = \texttt{let}\ x = d(y)\ \texttt{in}\ P_i'\ \texttt{else}\ P_i''$ and $Q_i = \texttt{let}\ x = d(y)\ \texttt{in}\ Q_i'\ \texttt{else}\ Q_i''$ for some P_i', P_i'', Q_i', Q_i''.
P_i reduces to either P_i' via the Let-In rule, or P_i'' via the Let-Else rule.
In addition

$$\Pi_i = \frac{x \notin \mathrm{dom}(\Gamma) \qquad \Gamma(y) = [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \vee \Gamma(y) <: \mathrm{key}^{l'}(T) \qquad \dfrac{\Pi''}{\Gamma \vdash P_i'' \sim Q_i'' \to C_i''}}{\Gamma \vdash P_i \sim Q_i \to C_i (= C_i'')}.$$

We have $\alpha = \tau$. By assumption we also have $\sigma_P(y) = \sigma_P^1(y)$ and $\sigma_Q(y) = \sigma_Q^1(y)$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$. Hence, by definition of the well-typedness of substitutions, there exists $c_y \subseteq c_\sigma$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P(y) \sim \sigma_Q(y) : [\![\tau_m^{l,a}\,;\,\tau_n^{l',a}]\!] \to c_y$, or $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P(y) \sim \sigma_Q(y) : \mathrm{key}^{l'}(T) \to c_y$ In the first case by Lemma 16, $\sigma_P(y) = m$ and $\sigma_Q(y) = n$. Since m, n are nonces, $d(m) \downarrow= d(n) \downarrow= \bot$, and we thus have $d(\sigma_P(y)) \downarrow= d(\sigma_Q(y)) \downarrow= \bot$. Similarly, in the second case, by Lemma 20, $\sigma_P(y)$ and $\sigma_Q(y)$ are both keys in $\mathcal{K}$, and thus $d(\sigma_P(y)) \downarrow= d(\sigma_Q(y)) \downarrow= \bot$.
Therefore the reduction rule applied to P_i can only be Let-Else, and P_i is reduced to P_i''. Since we also have $d(\sigma_Q(y)) \downarrow= \bot$, this rule can also be applied to reduce Q_i into Q_i''. This proves point a).
We therefore have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$. We choose $\Gamma' = \Gamma$.
Since the substitutions and typing environments are unchanged by the reduction, point b) clearly holds.
Moreover, Π'', and the fact that

$$\left[\!\!\left[(\cup_{\times\, j \neq i} C_j) \cup_\times C_i'' \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q} = \left[\!\!\left[(\cup_{\times\, j} C_j) \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q}$$

which is consistent by hypothesis, prove point c) and conclude this case.
– PIFL: then $P_i = \texttt{if}\ M = M'\ \texttt{then}\ P_i^\top\ \texttt{else}\ P_i^\bot$ and $Q_i = \texttt{if}\ N = N'\ \texttt{then}\ Q_i^\top\ \texttt{else}\ Q_i^\bot$ for some $Q_i^\top, Q_i^\bot$. P_i reduces to P_i' which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \frac{\dfrac{\Pi^\top}{\Gamma \vdash P_i^\top \sim Q_i^\top \to C_i^\top} \qquad \dfrac{\Pi^\bot}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C_i^\bot} \qquad \dfrac{\Pi}{\Gamma \vdash M \sim N : \mathtt{LL} \to c} \qquad \dfrac{\Pi'}{\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'}}{\Gamma \vdash P_i \sim Q_i \to C_i = (C_i^\top \cup C_i^\bot) \cup_\forall (c \cup c')}$$

We have $\alpha = \tau$. In addition, by hypothesis, $t\sigma_P = t\sigma_P^1$ for $t \in \{M, M'\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N, N'\}$.
Since $\Gamma \vdash M \sim N : \mathtt{LL} \to c$, by Lemma 24, there exists $c'' \subseteq [\![c]\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma_P \sim N\sigma_Q : \mathtt{LL} \to c''$. Similarly, there exists $c''' \subseteq [\![c']\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M'\sigma_P \sim N'\sigma_Q : \mathtt{LL} \to c'''$.
Hence, by Lemma 22, either $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$; or $M\sigma_P \downarrow= M\sigma_P \neq \bot$ and $N\sigma_Q \downarrow= N\sigma_Q \neq \bot$.
Similarly, either $M'\sigma_P \downarrow= N'\sigma_Q \downarrow= \bot$; or $M'\sigma_P \downarrow= M'\sigma_P \neq \bot$ and $N'\sigma_Q \downarrow= N'\sigma_Q \neq \bot$.
Let us first consider the case where $M\sigma_P \downarrow\neq \bot$, $M'\sigma_P \downarrow\neq \bot$, $N\sigma_Q \downarrow\neq \bot$ and $N'\sigma_Q \downarrow\neq \bot$.
Let $\phi = \{M\sigma_P/x, M'\sigma_P/y\}$ and $\phi' = \{N\sigma_Q/x, N'\sigma_Q/y\}$. We then have $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \phi \sim \phi' : \mathtt{LL} \to c'' \cup c'''$.
Let us prove that $c \cup c'''$ is consistent in some typing environment. By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$. Hence, by Lemma 6, there exists $\Gamma'' \in \mathrm{branches}(\Gamma)$ such that ${\Gamma''}_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : {\Gamma''}_{\mathcal{X}} \to c_\sigma$. By Lemma 15, there exists $(c_1, \Gamma''') \in C_i$ such that $\Gamma'' \subseteq \Gamma'''$. Since $C_i = (C_i^\top \cup C_i^\bot) \cup_\forall (c \cup c')$, c_1 is of the form $c_2 \cup c \cup c'$ for some c_2.
As we noted previously, $[\![C_i]\!]_{\sigma_P,\sigma_Q} \cup_\forall c_\sigma$ is consistent. Therefore, by Lemma 13, $\{([\![c \cup c']\!]_{\sigma_P,\sigma_Q} \cup c_\sigma, \Gamma''')\}$ is consistent. Hence, by the same Lemma, $c'' \cup c'''$ is also consistent in Γ'''.

Thus, by Lemma 27, ϕ and ϕ' are statically equivalent. Hence, in particular, $M\sigma_P = M'\sigma_P \iff N\sigma_Q = N'\sigma_Q$.
Therefore, if rule If-Then is applied to P_i then it can also be applied to reduce Q_i into $Q_i^\top$, and if the rule applied to P_i is If-Else then it can also be applied to reduce Q_i into $Q_i^\perp$. This proves point a). We prove here the If-Then case. The If-Else case is similar.
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and environments do not change in this reduction, point b) trivially holds.
Moreover, by hypothesis,

$$\left[\!\left[(\cup_{\times j\neq i} C_j)\cup_\times (C_i^\top \cup C_i^\perp)\cup_\forall (c \cup c' \cup c_\phi)\right]\!\right]_{\sigma_P,\sigma_Q}$$

is consistent. Thus by Lemma 13,

$$\left[\!\left[(\cup_{\times j\neq i} C_j)\cup_\times (C_i^\top \cup C_i^\perp)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

is also consistent. Since, using $C_i' = C_i^\top$ and $C_i = (C_i^\top \cup C_i^\perp)\cup_\forall (c \cup c')$,

$$\left[\!\left[(\cup_{\times j\neq i} C_j)\cup_\times C_i'\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times j\neq i} C_j)\cup_\times (C_i^\top \cup C_i^\perp)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q},$$

we have by Lemma 13 that $\left[\!\left[(\cup_{\times j\neq i} C_j)\cup_\times C_i'\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is consistent. $\Pi^\top$ and this fact prove point c) and conclude this case.

The case where $M\sigma_P \downarrow = N\sigma_Q \downarrow = \perp$ or $M'\sigma_P \downarrow = N'\sigma_Q \downarrow = \perp$ remains. In that case, the rule applied to P_i is necessarily $If - Else$, and this rule can also be applied to Q_i. We conclude the proof similarly to the previous case.

– PIFLR: then $P_i = \texttt{if } M_1 = M_2 \texttt{ then } P_i^\top \texttt{ else } P_i^\perp$ and $Q_i = \texttt{if } N_1 = N_2 \texttt{ then } Q_i^\top \texttt{ else } Q_i^\perp$ for some $Q_i^\top$, $Q_i^\perp$. P_i reduces to P_i' which is either $P_i^\top$ via the If-Then rule, or $P_i^\perp$ via the If-Else rule. In addition

$$\Pi_i = \frac{b = (\tau_m^{l,1} \stackrel{?}{=} \tau_{m'}^{l'',1}) \qquad b' = (\tau_n^{l',1} \stackrel{?}{=} \tau_{n'}^{l''',1}) \qquad \dfrac{\Pi}{\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,1}\,;\,\tau_n^{l',1}]\!] \to \emptyset} \qquad \dfrac{\Pi'}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',1}\,;\,\tau_{n'}^{l''',1}]\!] \to c'} \qquad \dfrac{\Pi''}{\Gamma \vdash P_i^b \sim Q_i^{b'} \to C_i'}}{\Gamma \vdash P_i \sim Q_i \to C_i = C_i'}$$

We have $\alpha = \tau$ in any case. In addition, by assumption, $t\sigma_P = t\sigma_P^1$ for $t \in \{M_1, M_2\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N_1, N_2\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_\mathcal{X} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists c'' such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M_1\sigma_P \sim N_1\sigma_Q : [\![\tau_m^{l,1}\,;\,\tau_n^{l',1}]\!] \to c''$. Therefore by Lemma 16, $M_1\sigma_P = m$ and $N_1\sigma_Q = n$. Similarly we can show that $M_2\sigma_P = m'$ and $N_2\sigma_Q = n'$.
There are four cases for b and b', which are all similar. We write the proof for the case where $b = \top$ and $b' = \perp$, *i.e.* $\tau_m^{l,1} = \tau_{m'}^{l'',1}$ and $\tau_n^{l',1} \neq \tau_{n'}^{l''',1}$.
Thus the reduction rule applied to P_i is If-Then and $P_i' = P_i^\top$. On the other hand, rule If-Else can be applied to reduce Q_i into $Q_i' = Q_i^\perp$. This proves point a) (these rules both correspond to silent actions).
We choose $\Gamma' = \Gamma$. We have $\sigma_P' = \sigma_P$ and $\sigma_Q' = \sigma_Q$.

Since the substitutions and environments do not change in this reduction, point b) trivially holds.
Moreover, Π'' and the fact that

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C'_i \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} = \left[\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

prove point c) and conclude this case.

– PIFS: then $P_i = \texttt{if}\ M = M'\ \texttt{then}\ P_i^\top\ \texttt{else}\ P_i^\bot$ and $Q_i = \texttt{if}\ N = N'\ \texttt{then}\ Q_i^\top\ \texttt{else}\ Q_i^\bot$ for some $Q_i^\top, Q_i^\bot$. P_i reduces to P'_i which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \frac{\dfrac{\Pi^\bot}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C'_i} \qquad \dfrac{\Pi}{\Gamma \vdash M \sim N : \texttt{LL} \to c} \qquad \dfrac{\Pi'}{\Gamma \vdash M' \sim N' : \texttt{HH} \to c'}}{\Gamma \vdash P_i \sim Q_i \to C_i = C'_i}$$

We have $\alpha = \tau$ in any case. In addition, by hypothesis, $t\sigma_P = t\sigma_P^1$ for $t \in \{M, M'\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N, N'\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_\mathcal{X} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists c'' such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma_P \sim N\sigma_Q : \texttt{LL} \to c''$. Similarly we can show that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M'\sigma_P \sim N'\sigma_Q : \texttt{HH} \to c'''$ for some c'''.
Hence, by Lemma 22, either $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$; or $M\sigma_P \downarrow= M\sigma_P \neq \bot$ and $N\sigma_Q \downarrow= N\sigma_Q \neq \bot$.
Similarly, either $M'\sigma_P \downarrow= N'\sigma_Q \downarrow= \bot$; or $M'\sigma_P \downarrow= M'\sigma_P \neq \bot$ and $N'\sigma_Q \downarrow= N'\sigma_Q \neq \bot$.
Let us first consider the case where $M\sigma_P \downarrow\neq \bot$, $M'\sigma_P \downarrow\neq \bot$, $N\sigma_Q \downarrow\neq \bot$ and $N'\sigma_Q \downarrow\neq \bot$.
Therefore by Lemma 25, $M\sigma_P \neq M'\sigma_P$ and $N\sigma_Q \neq N'\sigma_Q$. Hence the reduction for P_i is necessarily If-Else, which is also applicable to reduce Q_i to $Q_i^\bot$. This proves point a).
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and typing environments do not change in this reduction, point b) trivially holds.
Moreover, Π'' and the fact that

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C'_i \cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} = \left[\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

prove point c) and conclude this case.

The case where $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$ or $M'\sigma_P \downarrow= N'\sigma_Q \downarrow= \bot$ remains. In that case, the rule applied to P_i is necessarily $If - Else$, and this rule can also be applied to Q_i. We conclude the proof similarly to the previous case.

– PIFI: then $P_i = \texttt{if}\ M = M'\ \texttt{then}\ P_i^\top\ \texttt{else}\ P_i^\bot$ and $Q_i = \texttt{if}\ N = N'\ \texttt{then}\ Q_i^\top\ \texttt{else}\ Q_i^\bot$ for some $Q_i^\top, Q_i^\bot$. This case is similar to the PIFS case: the incompatibility of the types of M, N and M', N' ensures that the processes can only follow the else branch.
P_i reduces to P'_i which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \frac{\dfrac{\Pi^\bot}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C'_i} \qquad \dfrac{\Pi}{\Gamma \vdash M \sim N : T * T' \to c} \qquad \dfrac{\Pi'}{\Gamma \vdash M' \sim N' : [\![\tau_m^{l,a}\,;\, \tau_n^{l',a}]\!] \to c'}}{\Gamma \vdash P_i \sim Q_i \to C_i = C'_i}$$

We have $\alpha = \tau$ in any case. In addition, by hypothesis, $t\sigma_P = t\sigma_P^1$ for $t \in \{M, M'\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N, N'\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_\mathcal{X} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists c'' such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma_P \sim N\sigma_Q : T * T' \to c''$.

Hence, by Lemma 22, either $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$; or $M\sigma_P \downarrow= M\sigma_P \neq \bot$ and $N\sigma_Q \downarrow= N\sigma_Q \neq \bot$. Let us first consider the case where $M\sigma_P \downarrow\neq \bot$, and $N\sigma_Q \downarrow\neq \bot$.
By Lemma 19, $M\sigma_P$ and $N\sigma_Q$ both are pairs. Similarly we can show that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M'\sigma_P \sim N'\sigma_Q : [\![\tau_m^{l,a} ; \tau_n^{l',a}]\!] \to c'''$ for some c'''. By Lemma 16, this implies that $M'\sigma_P = m$ and $N'\sigma_Q = n$. Thus neither of these two terms are pairs.
Therefore $M\sigma_P \neq M'\sigma_P$ and $N\sigma_Q \neq N'\sigma_Q$. The end of the proof for this case is then the same as for the PIFS case.

The case where $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$ or $M'\sigma_P \downarrow= N'\sigma_Q \downarrow= \bot$ remains. In that case, the rule applied to P_i is necessarily $If - Else$, and this rule can also be applied to Q_i. We conclude the proof similarly to the previous case.

– PIFP: then $P_i = \texttt{if } M = t \texttt{ then } P_i^\top \texttt{ else } P_i^\bot$ and $Q_i = \texttt{if } N = t \texttt{ then } Q_i^\top \texttt{ else } Q_i^\bot$ for some $Q_i^\top$, $Q_i^\bot$, some messages M, N, and some $t \in C \cup \mathcal{K} \cup \mathcal{N}$. P_i reduces to P_i' which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \dfrac{\dfrac{\Pi}{\Gamma \vdash M \sim N : \mathtt{LL} \to c} \qquad \dfrac{\Pi'}{\Gamma \vdash t \sim t : \mathtt{LL} \to c'} \quad t \in \mathcal{C} \cup \mathcal{K} \cup \mathcal{N}}{\Gamma \vdash P_i \sim Q_i \to C_i = C_i^\top \cup C_i^\bot} \qquad \dfrac{\Pi^\top}{\Gamma \vdash P_i^\top \sim Q_i^\top \to C_i^\top} \qquad \dfrac{\Pi^\bot}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C_i^\bot}$$

We have in any case $\alpha = \tau$. In addition, by assumption, $t'\sigma_P = t'\sigma_P^1$ for $t' \in \{M, M'\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N, N'\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists $c'' \subseteq [\![c]\!]_{\sigma_P,\sigma_Q} \cup c_\sigma$ such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M\sigma_P \sim N\sigma_Q : \mathtt{LL} \to c''$.
Hence, by Lemma 22, either $M\sigma_P \downarrow= N\sigma_Q \downarrow= \bot$; or $M\sigma_P \downarrow= M\sigma_P \neq \bot$ and $N\sigma_Q \downarrow= N\sigma_Q \neq \bot$. Similarly, either $t \downarrow= \bot$ or $t \downarrow= t \neq \bot$.
Let us first consider the case where $M\sigma_P \downarrow\neq \bot$, $M'\sigma_P \downarrow\neq \bot$, and $t \downarrow\neq \bot$.
We then show that $M\sigma_P = t$ if and only if $N\sigma_Q = t$ (note that since t is ground, $t = t\sigma_P = t\sigma_Q$). If $M\sigma_P = t$, then $\Gamma_{\mathcal{N},\mathcal{K}} \vdash t \sim N\sigma_Q : \mathtt{LL} \to c''$. In all possible cases for t, *i.e.* $t \in \mathcal{K}$, $t \in \mathcal{N}$, and $t \in \mathcal{C}$, Lemma 20 implies that $N\sigma_Q = t$. This proves the first direction of the equivalence, the other direction is similar.
Therefore, if rule If-Then is applied to P_i then it can also be applied to reduce Q_i into $Q_i^\top$, and if the rule applied to P_i is If-Else then it can also be applied to reduce Q_i into $Q_i^\bot$. This proves point a). We prove here the If-Then case. The If-Else case is similar.
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and typing environments do not change in this reduction, point b) trivially holds.
Moreover, by hypothesis,

$$\left[\!\!\left[(\cup_{\times j\neq i} C_j) \cup_\times (C_i^\top \cup C_i^\bot) \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q}$$

is consistent. Since, using $C_i' = C_i^\top$ and $C_i = (C_i^\top \cup C_i^\bot)$, we have

$$\left[\!\!\left[(\cup_{\times j\neq i} C_j) \cup_\times C_i' \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\!\left[(\cup_{\times j\neq i} C_j) \cup_\times (C_i^\top \cup C_i^\bot) \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q},$$

we have by Lemma 13 that $\left[\!\!\left[(\cup_{\times j\neq i} C_j) \cup_\times C_i' \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q}$ is consistent. This fact proves point c) and concludes this case.

The case where $M\sigma_P \downarrow = N\sigma_Q \downarrow = \bot$ or $t \downarrow = \bot$ remains. In that case, the rule applied to P_i is necessarily $If - Else$, and this rule can also be applied to Q_i. We conclude the proof similarly to the previous case.

– PIfLR*: then $P_i = \texttt{if } M_1 = M_2 \texttt{ then } P_i^\top \texttt{ else } P_i^\bot$ and $Q_i = \texttt{if } N_1 = N_2 \texttt{ then } Q_i^\top \texttt{ else } Q_i^\bot$ for some $Q_i^\top$, $Q_i^\bot$. P_i reduces to P_i' which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \dfrac{\dfrac{\Pi'}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_m^{l,\infty}\,;\tau_n^{l',\infty}]\!] \to c_2} \quad \dfrac{\Pi^\top}{\Gamma \vdash P_i^\top \sim Q_i^\top \to C_i^\top} \quad \dfrac{\Pi^\bot}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C_i^\bot}}{\Gamma \vdash P_i \sim Q_i \to C_i = C_i^\top \cup C_i^\bot}$$

We have $\alpha = \tau$ in any case. In addition, by assumption, $t\sigma_P = t\sigma_P^1$ for $t \in \{M_1, M_2\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N_1, N_2\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists c'' such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M_1\sigma_P \sim N_1\sigma_Q : [\![\tau_m^{l,\infty}\,;\tau_m^{l',\infty}]\!] \to c''$. Therefore by Lemma 16, $M_1\sigma_P = m$ and $N_1\sigma_Q = n$. Similarly we can show that $M_2\sigma_P = m$ and $N_2\sigma_Q = n$.
Hence $M_1' = M_2'$ and $N_1' = N_2'$.
Thus the reduction rule applied to P_i is If-Then and $P_i' = P_i^\top$. On the other hand, rule If-Then can also be applied to reduce Q_i into $Q_i' = Q_i^\top$. This proves point a).
Note that we still need to type the other branch, even though it is not used here, as when replicating the process this test may fail if M_1, N_1 and M_2, N_2 are nonces from different sessions.
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and environments do not change in this reduction, point b) trivially holds.
Moreover, Π'' and the fact that, with $C_i' = C_i^\top$,

$$\begin{aligned}\left[\!\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i' \cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q} &\subseteq \left[\!\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times (C_i^\top \cup C_i^\bot)\cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q} \\ &= \left[\!\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\!\right]_{\sigma_P,\sigma_Q}\end{aligned}$$

prove point c) and conclude this case.

– PIfLR'*: then $P_i = \texttt{if } M_1 = M_2 \texttt{ then } P_i^\top \texttt{ else } P_i^\bot$ and $Q_i = \texttt{if } N_1 = N_2 \texttt{ then } Q_i^\top \texttt{ else } Q_i^\bot$ for some $Q_i^\top$, $Q_i^\bot$. P_i reduces to P_i' which is either $P_i^\top$ via the If-Then rule, or $P_i^\bot$ via the If-Else rule. In addition

$$\Pi_i = \dfrac{\dfrac{\Pi'}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',a}\,;\tau_{n'}^{l''',a}]\!] \to c_2} \quad \tau_m^{l,a} \neq \tau_{m'}^{l'',a} \quad \tau_n^{l',a} \neq \tau_{n'}^{l''',a} \quad \dfrac{\Pi''}{\Gamma \vdash P_i^\bot \sim Q_i^\bot \to C_i'}}{\Gamma \vdash P_i \sim Q_i \to C_i = C_i'}$$

We have $\alpha = \tau$ in any case. In addition, by assumption, $t\sigma_P = t\sigma_P^1$ for $t \in \{M_1, M_2\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N_1, N_2\}$.
By hypothesis, σ_P, σ_Q are ground and $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma_P \sim \sigma_Q : \Gamma_{\mathcal{X}} \to c_\sigma$. Hence, by Lemma 24, using Π, there exists c'' such that $\Gamma_{\mathcal{N},\mathcal{K}} \vdash M_1\sigma_P \sim N_1\sigma_Q : [\![\tau_m^{l,a}\,;\tau_n^{l',a}]\!] \to c''$. Therefore by Lemma 16, $M_1\sigma_P = m$ and $N_1\sigma_Q = n$. Similarly, using Lemma 16, we can show that $M_2\sigma_P = m'$ and $N_2\sigma_Q = n'$.

Moreover, since $\tau_m^{l,a} \neq \tau_{m'}^{l'',a}$, we know that $m \neq m'$ (by well-formedness of the processes), and similarly $n \neq n'$.
Hence, $M_1\sigma_P \neq M_2\sigma_P$ and $N_1\sigma_Q \neq N_2\sigma_Q$.
Thus the reduction rule applied to P_i is If-Else and $P_i' = P_i^{\perp}$. On the other hand, rule If-Else can also be applied to reduce Q_i into $Q_i' = Q_i^{\perp}$. This proves point a).
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and environments do not change in this reduction, point b) trivially holds.
Moreover, Π'' and the fact that

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} = \left[\!\left[(\cup_{\times\, j} C_j)\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

prove point c) and conclude this case.

– PIFALL: then $P_i = \texttt{if } M = M' \texttt{ then } P_i^{\top} \texttt{ else } P_i^{\perp}$ and $Q_i = \texttt{if } N = N' \texttt{ then } Q_i^{\top} \texttt{ else } Q_i^{\perp}$ for some $Q_i^{\top}, Q_i^{\perp}$. P_i reduces to P_i' which is either $P_i^{\top}$ via the If-Then rule, or $P_i^{\perp}$ via the If-Else rule. In addition

$$\Pi_i = \dfrac{\dfrac{\Pi^{\top,\bot}}{\Gamma \vdash P_i^{\top} \sim Q_i^{\bot} \to C_i^{\top,\bot}} \quad \dfrac{\Pi^{\bot,\top}}{\Gamma \vdash P_i^{\bot} \sim Q_i^{\top} \to C_i^{\bot,\top}} \quad \dfrac{\Pi^{\bot,\bot}}{\Gamma \vdash P_i^{\bot} \sim Q_i^{\bot} \to C_i^{\bot,\bot}}}{\begin{array}{c}\Gamma \vdash \texttt{if } M = M' \texttt{ then } P_i^{\top} \texttt{ else } P_i^{\perp} \sim \texttt{if } N = N' \texttt{ then } Q_i^{\top} \texttt{ else } Q_i^{\perp} \to \\ C_i = C_i^{\top,\top} \cup C_i^{\top,\bot} \cup C_i^{\bot,\top} \cup C_i^{\bot,\bot}\end{array}}$$

In addition, by hypothesis, $t\sigma_P = t\sigma_P^1$ for $t \in \{M, M'\}$ and $t'\sigma_Q = t'\sigma_Q^1$ for $t' \in \{N, N'\}$.
Four cases are possible:

- $M\sigma_P \downarrow \neq \perp$, $M'\sigma_P \downarrow \neq \perp$, $N\sigma_Q \downarrow \neq \perp$, $N'\sigma_Q \downarrow \neq \perp$, $M\sigma_P = M'\sigma_P$ and $N\sigma_Q = N'\sigma_Q$;
- or $M\sigma_P \downarrow \neq \perp$, $M'\sigma_P \downarrow \neq \perp$, $M\sigma_P = M'\sigma_P$ and ($N\sigma_Q \neq N'\sigma_Q$ or $N\sigma_Q \downarrow = \perp$ or $N'\sigma_Q \downarrow = \perp$);
- or $N\sigma_Q \downarrow \neq \perp$, $N'\sigma_Q \downarrow \neq \perp$, $N\sigma_Q = N'\sigma_Q$ and ($M\sigma_P \neq M'\sigma_P$ or $M\sigma_P \downarrow = \perp$ or $M'\sigma_P \downarrow = \perp$);
- or ($M\sigma_P \neq M'\sigma_P$ or $M\sigma_P \downarrow = \perp$ or $M'\sigma_P \downarrow = \perp$) and ($N\sigma_Q \neq N'\sigma_Q$ or $N\sigma_Q \downarrow = \perp$ or $N'\sigma_Q \downarrow = \perp$).

In any case, we have $\alpha = \tau$. These four cases are similar, we detail the proof for the second case, where $M\sigma_P = M'\sigma_P$ and $N\sigma_Q \neq N'\sigma_Q$.
Since $M\sigma_P = M'\sigma_P$, $M'\sigma_P \downarrow \neq \perp$, and $M\sigma_P = M'\sigma_P$, the reduction applied to P_i can only be If-Then, and P_i is reduced to $P_i^{\top}$. Since $N\sigma_Q \neq N'\sigma_Q$, $N\sigma_Q \downarrow = \perp$, or $N'\sigma_Q \downarrow = \perp$, rule If-Else can be applied to reduce Q_i into $Q_i^{\perp}$. This proves point a).
We choose $\Gamma' = \Gamma$. We have $\sigma_P^2 = \sigma_P^1$ and $\sigma_Q^2 = \sigma_Q^1$.
Since the substitutions and environments do not change in this reduction, point b) trivially holds.
Moreover, by hypothesis,

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times (C_i^{\top,\top} \cup C_i^{\top,\bot} \cup C_i^{\bot,\top} \cup C_i^{\bot,\bot})\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$$

is consistent. Since, using $C_i' = C_i^{\top,\bot}$,

$$\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q} \subseteq \left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times (C_i^{\top,\top} \cup C_i^{\top,\bot} \cup C_i^{\bot,\top} \cup C_i^{\bot,\bot})\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q},$$

we have by Lemma 13 that $\left[\!\left[(\cup_{\times\, j\neq i} C_j)\cup_\times C_i'\cup_\forall c_\phi\right]\!\right]_{\sigma_P,\sigma_Q}$ is consistent. $\Pi^{\top,\bot}$ and this fact prove point c) and conclude this case.

Theorem 4 (Typing implies trace inclusion). *For all processes P, Q, for all ϕ_P, ϕ_Q, σ_P, σ_Q, for all multisets of processes $\mathcal{P}$, $\mathcal{Q}$ for all constraints C, for all sequence s of actions, for all Γ containing only keys, if*

$$\Gamma \vdash P \sim Q \to C,$$

and if C is consistent, then

$$P \sqsubseteq_t Q$$

that is, if

$$(\{P\}, \emptyset, \emptyset) \xrightarrow{s}_* (\mathcal{P}, \phi_P, \sigma_P),$$

then there exists a sequence s' of actions, a multiset $\mathcal{Q}$, a frame ϕ_Q, a substitution σ_Q, such that

- $s =_\tau s'$
- $(\{Q\}, \emptyset, \emptyset) \xrightarrow{s'}_* (\mathcal{Q}, \phi_Q, \sigma_Q)$,
- *$\phi_P\sigma_P$ and $\phi_Q\sigma_Q$ are statically equivalent.*

Proof. We successively apply Lemma 28 to each of the reduction steps in the reduction

$$(\{P\}, \emptyset, \emptyset) \xrightarrow{s}_* (\mathcal{P}, \phi_P, \sigma_P).$$

The lemma can indeed be applied successively. At each reduction step of P we obtain a sequence of reduction steps for Q with the same actions, and the conclusions the lemma provides imply the conditions needed for its next application.

It is clear, for the first application, that all the hypotheses of this lemma are satisfied.

In the end, we know that there exist Γ', some constraint sets C_i, some c_ϕ, c_σ, and a reduction

$$(\{Q\}, \emptyset, \emptyset) \xrightarrow{s'}_* (\mathcal{Q}, \phi_Q, \sigma_Q)$$

with $s =_\tau s'$, such that (among other conclusions)

- σ_P, σ_Q are ground, and there exist ground $\sigma'_P \supseteq \sigma_P$, $\sigma'_Q \supseteq \sigma_Q$ such that
  - $(\mathrm{dom}(\sigma'_P) \backslash \mathrm{dom}(\sigma_P)) \cap (\mathrm{vars}(\mathcal{P}) \cup \mathrm{vars}(\phi_P)) = \emptyset$,
  - $(\mathrm{dom}(\sigma'_Q) \backslash \mathrm{dom}(\sigma_Q)) \cap (\mathrm{vars}(\mathcal{Q}) \cup \mathrm{vars}(\phi_Q)) = \emptyset$, and
  - $\Gamma_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P \sim \sigma'_Q : \Gamma_{\mathcal{X}} \to c_\sigma$,
  - for all $x \in \mathrm{dom}(\sigma'_P)$, $\sigma'_P(x) \downarrow = \sigma'_P(x)$, and similarly for σ'_Q,
- $c_\sigma \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$,
- $\Gamma' \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$,
- $\mathrm{dom}(\phi_P) = \mathrm{dom}(\phi_Q)$,
- $\forall i, \Gamma' \vdash P_i \sim Q_i \to C_i$,
- for all $i \neq j$, the sets of bound variables in P_i and P_j (resp. Q_i and Q_j) are disjoint, and similarly for the bound names;
- $[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q}$ is consistent.

To prove the claim, it is then sufficient to show that $\phi_P\sigma_P$ and $\phi_Q\sigma_Q$ are statically equivalent.

Note that since $\sigma_P \subseteq \sigma'_P$ and $(\mathrm{dom}(\sigma'_P) \backslash \mathrm{dom}(\sigma_P)) \cap (\mathrm{vars}(\mathcal{P}) \cup \mathrm{vars}(\phi_P)) = \emptyset$, we have $\phi_P\sigma_P = \phi_P\sigma'_P$. Similarly $\phi_Q\sigma_Q = \phi_Q\sigma'_Q$. In addition we also know that $[\![c_\phi]\!]_{\sigma_P,\sigma_Q} = [\![c_\phi]\!]_{\sigma'_P,\sigma'_Q}$.

We have $\Gamma' \vdash \phi_P \sim \phi_Q : \mathtt{LL} \to c_\phi$ and ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma'_P \sim \sigma'_Q : {\Gamma'}_{\mathcal{X}} \to c_\sigma$. Hence, by Lemma 24, there exists $c \subseteq [\![c_\phi]\!]_{\sigma'_P,\sigma'_Q} \cup c_\sigma$ (*i.e.* such that $c \subseteq [\![c_\phi]\!]_{\sigma'_P,\sigma'_Q}$) such that ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \phi_P\sigma'_P \sim \phi_Q\sigma'_Q : \mathtt{LL} \to c$.

We will now show that $(c, \Gamma'_{\mathcal{N},\mathcal{K}})$ is consistent.

Let $\Gamma'' \in \mathrm{branches}(\Gamma')$. By Lemma 14, for all i, since $\Gamma' \vdash P_i \sim Q_i \to C_i$, there exists $(c_i, \Gamma_i''') \in C_i$ such that $\Gamma'' \subseteq \Gamma_i''$. The disjointness condition on the bound variables implies by Lemma 10 that for all i, j, Γ_i''' and Γ_j''' are compatible. Thus $\cup_{\times i} C_i$ contains $(c', \Gamma''') \stackrel{\text{def}}{=} (\bigcup_i c_i, \bigcup_i \Gamma_i''')$. We have $\Gamma'' \subseteq \Gamma'''$. Therefore $[\![(\cup_{\times i} C_i) \cup_\forall c_\phi]\!]_{\sigma_P,\sigma_Q}$, which is consistent, contains $([\![c' \cup c_\phi]\!]_{\sigma_P,\sigma_Q}, \Gamma''')$. Therefore, as $c \subseteq [\![c_\phi]\!]_{\sigma_P,\sigma_Q}$, by Lemma 13, (c, Γ''') is consistent. Since c is ground, it follows from the definition of consistency that $(c, \Gamma'''_{\mathcal{N},\mathcal{K}})$ is also consistent. Moreover, we know that $\Gamma'' \subseteq \Gamma'''$, and Γ'' is a branch of Γ'. It is then clear that $\Gamma'_{\mathcal{N},\mathcal{K}} \subseteq \Gamma'''$. Hence, by Lemma 12, since $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \phi_P\sigma_P \sim \phi_Q\sigma_Q : \mathtt{LL} \to c$, we have $\Gamma''' \vdash \phi_P\sigma_P \sim \phi_Q\sigma_Q : \mathtt{LL} \to c$.

Hence, we have $\Gamma''' \vdash \phi_P\sigma_P \sim \phi_Q\sigma_Q : \mathtt{LL} \to c$ with $(c, \Gamma'''_{\mathcal{N},\mathcal{K}})$ consistent.
Moreover, $\phi_P\sigma_P$ and $\phi_Q\sigma_Q$ are ground (by well-formedness of the processes).
Therefore, by Lemma 27, the frames $\phi_P\sigma_P$ and $\phi_Q\sigma_Q$ are statically equivalent.

This theorem corresponds to Theorem 1.

Theorem 5 (Typing implies trace equivalence). *For all Γ containing only keys, for all P and Q, if*

$$\Gamma \vdash P \sim Q \to C$$

and C is consistent, then

$$P \approx_t Q.$$

Proof. Theorem 4 proves that under these assumptions, $P \sqsubseteq_t Q$. This is sufficient to prove the theorem. Indeed, it is clear from the typing rules for processes and terms that

$$\Gamma \vdash P \sim Q \to C \Leftrightarrow \Gamma' \vdash Q \sim P \to C'$$

where C' is the constraint obtained from C by swapping the left and right hand sides of all of its elements, and Γ' is the environment obtained from Γ by swapping the left and right types in all refinement types, as well as swapping all pairs of keys in its domain. Clearly from the definition of consistency, C is consistent if and only if C' is. Therefore, by symmetry, proving that the assumptions imply $P \sqsubseteq_t Q$ also proves that they imply $Q \sqsubseteq_t P$, and thus $P \approx_t Q$.

B.2 Typing replicated processes

In this subsection, we prove the soundness result for replicated processes.

In this subsection, as well as the following ones, without loss of generality we assume, for each infinite nonce type $\tau_m^{l,\infty}$ appearing in the processes we consider, that $\mathcal{N}$ contains an infinite number of fresh names which we will denote by $\{m_i \mid i \in \mathbb{N}\}$; such that the m_i do not appear in the processes or environments considered. We will denote by $\mathcal{N}_0$ the set of unindexed names and by $\mathcal{N}_i$ the set of indexed names. We similarly assume that for all the variables x appearing in the processes, the set $\mathcal{X}$ of all variables also contains variables $\{x_i \mid i \in \mathbb{N}\}$. We denote $\mathcal{X}_0$ the set of unindexed variables, and $\mathcal{X}_i$ the set of indexed variables. Finally, we assume for all key k declared in the processes with type $\mathrm{seskey}^{l,\infty}(T)$ that the set $\mathcal{BK}$ contains keys $\{k_i \mid i \in \mathbb{N}\}$.

Definition 17 (Renaming of a term). *We denote by $[\, t \,]_i^\Gamma$, the term t in which names n such that $\Gamma(n) = \tau_n^{l,\infty}$ for some l are replaced by n_i, keys k such that $\Gamma(k,k) = \mathrm{seskey}^{l,\infty}(T)$ for some l, T are replaced by k_i, and variables x are replaced by x_i.*

Definition 18 (Expansion of a type). *given a type T, we define its expansion to n sessions, denoted $[\,T\,]^n$, as follows.*

$$
\begin{aligned}
[\,l\,]^n &= l \\
[\,T * T'\,]^n &= [\,T\,]^n * [\,T'\,]^n \\
\left[\,\mathrm{key}^l(T)\,\right]^n &= \mathrm{key}^l([\,T\,]^n) \\
\left[\,\mathrm{eqkey}^l(T)\,\right]^n &= \mathrm{eqkey}^l([\,T\,]^n) \\
\left[\,\mathrm{seskey}^{l,a}(T)\,\right]^n &= \mathrm{seskey}^{l,1}([\,T\,]^n) \\
[\,\mathrm{pkey}(T)\,]^n &= \mathrm{pkey}([\,T\,]^n) \\
[\,\mathrm{vkey}(T)\,]^n &= \mathrm{vkey}([\,T\,]^n) \\
[\,(T)_{T'}\,]^n &= ([\,T\,]^n)_{[\,T'\,]^n} \\
[\,\{T\}_{T'}\,]^n &= \{[\,T\,]^n\}_{[\,T'\,]^n} \\
[\,T \vee T'\,]^n &= [\,T\,]^n \vee [\,T'\,]^n \\
\left[\,[\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!]\,\right]^n &= [\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!] \\
\left[\,[\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]\,\right]^n &= \bigvee_{j=1}^{n} [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]
\end{aligned}
$$

where $l, l' \in \{\mathtt{LL}, \mathtt{HH}, \mathtt{HL}\}$, $k \in \mathcal{K}$. Note that the size of the expanded type $[\,T\,]^n$ depends on n.

Definition 19 (Renaming of a process). *For all process P, for all $i \in \mathbb{N}$, for all environment Γ, we define $[\,P\,]_i^\Gamma$, the renaming of P for session i with respect to Γ, as the process obtained from P by:*

- *for each nonce n declared in P by* `new` $n : \tau_n^{l,\infty}$*, and each nonce n such that $\Gamma(n) = \tau_n^{l,\infty}$ for some l, replacing every occurrence of n with n_i, and the declaration* `new` $n : \tau_n^{l,\infty}$ *with* `new` $n_i : \tau_{n_i}^{l,1}$*;*
- *for each key k declared in P by* `new` $k : \mathrm{seskey}^{l,\infty}(T)$*, and each key k such that $\Gamma(k,k) = \mathrm{seskey}^{l,\infty}(T)$ for some l, T replacing every occurrence of k with k_i, and the declaration* `new` $k : \mathrm{seskey}^{l,\infty}(T)$ *with* `new` $k_i : \mathrm{seskey}^{l,1}([\,T\,]^n)$*;*
- *replacing every occurence of a variable x with x_i.*

Definition 20 (Renaming and expansion of typing environments). *For any typing environment Γ, we define its renaming for session i as:*

$$
\begin{aligned}
[\,\Gamma\,]_i = \{&x_i : T \mid \Gamma(x) = T\} \\
&\cup \{(k,k') : T \mid \Gamma(k,k') = T \wedge \forall l, T'. T \neq \mathrm{seskey}^{l,\infty}(T')\} \\
&\cup \{(k_i,k_i) : \mathrm{seskey}^{l,1}(T) \mid \Gamma(k,k) = \mathrm{seskey}^{l,\infty}(T) \\
&\cup\ \{m : \tau_m^{l,1} \mid \Gamma(m) = \tau_m^{l,1}\} \\
&\cup\ \{m_i : \tau_{m_i}^{l,1} \mid \Gamma(m) = \tau_m^{l,\infty}\}.
\end{aligned}
$$

and then its expansion to n sessions as

$$
\begin{aligned}
[\,\Gamma\,]_i^n = \{&x_i : [\,T\,]^n \mid [\,\Gamma\,]_i(x_i) = T\} \cup \{k : [\,T\,]^n \mid [\,\Gamma\,]_i(k) = T\} \\
&\cup\ \{m : \tau_m^{l,1} \mid [\,\Gamma\,]_i(m) = \tau_m^{l,1}\}.
\end{aligned}
$$

Note that in $[\,\Gamma\,]_i^n$, due to the expansion, the size of the types depends on n.

Definition 21 (Renaming and expansion of constraints). *Given a set of constraints c, and an environment Γ, we define the renaming of c for session i in Γ as $[\,c\,]_i^\Gamma = \{[\,u\,]_i^\Gamma \sim [\,v\,]_i^\Gamma \mid u \sim v \in c\}$.*

This is propagated to constraint sets as follows: the renaming of C for session i is $[\,C\,]_i = \{([\,c\,]_i^\Gamma, [\,\Gamma\,]_i) \mid (c,\Gamma) \in C\}$ and its expansion to n sessions is $[\,C\,]_i^n = \{([\,c\,]_i^\Gamma, \Gamma') \mid \exists \Gamma.\ (c,\Gamma) \in C\ \wedge\ \Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)\}$.

Lemma 29 (Typing terms with replicated names). *For all Γ, M, N, T and c, if*

$$\Gamma \vdash M \sim N : T \to c$$

then for all $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$,

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\,T\,]^n \to [\,c\,]_i^\Gamma$$

Proof. Let Γ, M, N, T, c be such as assumed in the statement of the lemma. Let $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$. Let $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$.

We prove this property by induction on the proof Π of

$$\Gamma \vdash M \sim N : T \to c.$$

There are several possible cases for the last rule applied in Π.

- TNONCE: then $M = m$ and $N = p$ for some $m, p \in \mathcal{N}$, $T = l$ for some $l \in \{\mathtt{HH}, \mathtt{HL}\}$, and

$$\Pi = \frac{\Gamma(m) = \tau_m^{l,a} \qquad \Gamma(p) = \tau_p^{l,a}}{\Gamma \vdash m \sim p : l \to \emptyset}.$$

  It is clear from the definition of $[\,\Gamma\,]_i^n$ that $[\,\Gamma\,]_i^n([\,m\,]_i^\Gamma) = \tau^{l,1}_{[\,m\,]_i^\Gamma}$, and that $[\,\Gamma\,]_i^n([\,p\,]_i^\Gamma) = \tau^{l,1}_{[\,p\,]_i^\Gamma}$. Hence, $\Gamma'([\,m\,]_i^\Gamma) = \tau^{l,1}_{[\,m\,]_i^\Gamma}$ and $\Gamma'([\,p\,]_i^\Gamma) = \tau^{l,1}_{[\,p\,]_i^\Gamma}$. Then, by rule TNONCE, we have $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : l \to \emptyset$ and the claim holds.
- TNONCEL, TCSTFN, TKEY, TPUBKEYL, TVKEYL, THIGH, TLR[1]: Similarly to the TNONCE case, the claim follows directly from the definition of $[\,\Gamma\,]_i^n$, $[\,M\,]_i^\Gamma$, $[\,N\,]_i^\Gamma$, $[\,T\,]^n$ and $[\,c\,]_i^\Gamma$ in these cases.
- TENCH: then $T = \mathtt{LL}$ and there exist T', k, c' such that

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash M \sim N : (T')_{T''} \to c'} \qquad T'' <: \mathrm{key}^{\mathtt{HH}}(T')}{\Gamma \vdash M \sim N : \mathtt{LL} \to c = c' \cup \{M \sim N\}}.$$

  By applying the induction hypothesis to Π', since $[\,(T')_{T''}\,]^n = ([\,T'\,]^n)_{[\,T''\,]^n}$, there exists a proof Π'' of $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : ([\,T'\,]^n)_{[\,T''\,]^n} \to [\,c'\,]_i^\Gamma$
  In addition it is clear by definition of $[\,\cdot\,]^n$ and Lemma 3 that since $T'' <: \mathrm{key}^{\mathtt{HH}}(T')$ we have $[\,T''\,]^n <: \mathrm{key}^{\mathtt{HH}}([\,T'\,]^n)$.
  Therefore by rule TENCH, we have

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \mathtt{LL} \to [\,c'\,]_i^\Gamma \cup \{[\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma\} = [\,c\,]_i^\Gamma$$

- TPAIR, TPUBKEY, TVKEY, TENC, TENCL, TAENC, TAENCH, TAENCL, TSIGNH, TSIGNL as well as TOR,THASH, THASHL: Similarly to the TENCH case, the claim is proved directly by applying the induction hypothesis to the type judgement appearing in the conditions of the last rule in these cases.
- TVAR: then $M = N = x$ for some $x \in \mathcal{X}$, and

$$\Pi = \frac{\Gamma(x) = T}{\Gamma \vdash x \sim x : T \to \emptyset}.$$

  We have $[\,M\,]_i^\Gamma = [\,N\,]_i^\Gamma = x_i$.

Since $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, we have $\Gamma'(x_i) \in \mathrm{branches}([\,T\,]^n)$.
Hence by rule TVAR, $\Gamma' \vdash x_i \sim x_i : \Gamma'(x_i) \to \emptyset$. Therefore, by rule TOR, we have

$$\Gamma' \vdash x_i \sim x_i : [\,T\,]^n \to \emptyset$$

which proves the claim.

- TLR' (the TLRL' case is similar): then there exist m, p, l such that $T = l$, and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash M \sim N : [\![\tau_m^{l,a}\,;\,\tau_p^{l,a}]\!] \to c}}{\Gamma \vdash M \sim N : l \to c}.$$

Let us distinguish the case where a is 1 from the case where a is ∞.
If a is 1: by applying the induction hypothesis to Π', since $\big[\,[\![\tau_m^{l,a}\,;\,\tau_p^{l,a}]\!]\,\big]^n = [\![\tau_m^{l,1}\,;\,\tau_p^{l,1}]\!]$, we have

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\![\tau_m^{l,a}\,;\,\tau_p^{l,a}]\!] \to [\,c\,]_i^\Gamma.$$

Thus by rule TLR', we have

$$[\,\Gamma\,]_i^n \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : l \to [\,c\,]_i^\Gamma.$$

If a is ∞: by applying the induction hypothesis to Π', since

$$\big[\,[\![\tau_m^{l,a}\,;\,\tau_p^{l,a}]\!]\,\big]^n = \bigvee_{1\le j\le n} [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l,1}]\!],$$

we have

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \bigvee_{1\le j\le n} [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l,1}]\!] \to [\,c\,]_i^\Gamma.$$

Thus, by Lemma 7, there exists $j \in [\![1, n]\!]$ and a proof Π'' of

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l,1}]\!] \to [\,c\,]_i^\Gamma.$$

Thus, by rule TLR',

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : l \to [\,c\,]_i^\Gamma,$$

which proves the claim.

- TLRVAR: this case is similar to the TLR' case, but only the case where a is 1 is possible.
- TSUB: then there exists $T' <: T$ such that

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash M \sim N : T' \to c} \qquad T' <: T}{\Gamma \vdash M \sim N : T \to c}.$$

By applying the induction hypothesis to Π', we have

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\,T'\,]^n \to [\,c\,]_i^\Gamma.$$

Since it is clear by induction on the subtyping rules that $T' <: T$ implies that $[\,T'\,]^n <: [\,T\,]^n$, the TSUB rule can be applied and proves the claim.

– TLR$^\infty$: then $M = m$, $N = p$, $c = \emptyset$, and $T = [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]$ for some $m, p \in \mathcal{N}$, $c = \emptyset$, and

$$\Pi = \frac{\Gamma(m) = \tau_m^{l,\infty} \qquad \Gamma(p) = \tau_p^{l',\infty}}{\Gamma \vdash m \sim p : [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!] \to \emptyset}.$$

We have by definition $[\,M\,]_i^\Gamma = m_i$ and $[\,N\,]_i^\Gamma = p_i$, and $[\,\Gamma\,]_i^n(m_i) = \tau_{m_i}^{l,1}$, and $[\,\Gamma\,]_i^n(p_i) = \tau_{p_i}^{l',1}$. Thus $\Gamma'(m_i) = \tau_{m_i}^{l,1}$, and $\Gamma'(p_i) = \tau_{p_i}^{l',1}$. Hence by rule TLR[1], we have $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\![\tau_{m_i}^{l,1}\,;\,\tau_{p_i}^{l',1}]\!] \to \emptyset$.
In addition, $\left[\,[\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]\,\right]^n = \bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$. Therefore, by applying rule TOR, we have

$$\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \left[\,[\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]\,\right]^n \to \emptyset$$

which proves the claim.

Lemma 30 (Typing destructors with replicated names). *For all Γ, t, t', T, if*

$$\Gamma \vdash_d t \sim t' : T$$

then for all $i, n \in \mathbb{N}$ such that $1 \le i \le n$,

$$[\,\Gamma\,]_i^n \vdash_d [\,t\,]_i^\Gamma \sim [\,t'\,]_i^\Gamma : [\,T\,]^n$$

Proof. Immediate by examining the typing rules for destructors.

Lemma 31 (Branches and expansion).

– *For all T,*

$$\bigcup_{T' \in \mathrm{branches}(T)} \mathrm{branches}([\,T'\,]^n) = \mathrm{branches}([\,T\,]^n)$$

– *For all Γ for all $i, n \in \mathbb{N}$,*

$$\bigcup_{\Gamma' \in \mathrm{branches}(\Gamma)} \mathrm{branches}([\,\Gamma'\,]_i^n) = \mathrm{branches}([\,\Gamma\,]_i^n)$$

Proof. The first point is proved by induction on T.
If $T = T' \vee T''$ for some T', T'', then

$$\begin{aligned}\mathrm{branches}([\,T\,]^n) &= \mathrm{branches}([\,T'\,]^n) \cup \mathrm{branches}([\,T''\,]^n)\\ &= (\textstyle\bigcup_{T''' \in \mathrm{branches}(T')} \mathrm{branches}([\,T'''\,]^n)) \cup (\bigcup_{T''' \in \mathrm{branches}(T'')} \mathrm{branches}([\,T'''\,]^n))\end{aligned}$$

by the induction hypothesis. Since $\mathrm{branches}(T) = \mathrm{branches}(T') \cup \mathrm{branches}(T'')$, this proves the claim.
Otherwise, $\mathrm{branches}(T) = \{T\}$ and the claim trivially holds.

The second point directly follows from the first point, using the definition of $[\,\Gamma\,]_i^n$.

Lemma 32 (Typing processes in all branches). *For all P, Q, Γ, $\{C_{\Gamma'}\}_{\Gamma' \in \mathrm{branches}(\Gamma)}$, if*

$$\forall \Gamma' \in \mathrm{branches}(\Gamma). \quad \Gamma' \vdash P \sim Q \to C_{\Gamma'}$$

then

$$\Gamma \vdash P \sim Q \to \bigcup_{\Gamma' \in \mathrm{branches}(\Gamma)} C_{\Gamma'}.$$

Consequently if for some C, $C_{\Gamma'} \subseteq C$ for all Γ', then there exists $C' \subseteq C$ such that

$$\Gamma \vdash P \sim Q \to C'.$$

Proof. The first point is easily proved by successive applications of rule POR.

The second point is a direct consequence of the first point.

Lemma 33 (Expansion and union).

- *For all C, C', such that $\forall (c, \Gamma) \in C \cup C'.\ \mathrm{branches}(\Gamma) = \{\Gamma\}$,* i.e. *such that Γ does not contain union types, and $\mathrm{names}(c) \subseteq \mathrm{dom}(\Gamma) \cup \mathcal{FN}$, and Γ only nonce types with names from $\mathcal{N}_0$ (*i.e. *unindexed names), we have*
$$[\, C \cup_\times C' \,]^n_i = [\, C \,]^n_i \cup_\times [\, C' \,]^n_i$$
- *For all C, c, Γ, such that $\mathrm{names}(c) \subseteq \mathrm{dom}(\Gamma)$ and $\forall (c, \Gamma') \in C.\ \Gamma_{\mathcal{N},\mathcal{K}} \subseteq \Gamma'$, we have*
$$[\, C \cup_\forall c \,]^n_i = [\, C \,]^n_i \cup_\forall [\, c \,]^\Gamma_i$$

Proof. The first point follows from the definition of $[\,\cdot\,]^n_i$ and $\cup_\times$. Indeed, if C, C' are as assumed in the claim, we have:

$$\begin{aligned}
[\, C \cup_\times C' \,]^n_i &= \{([\, c \,]^\Gamma_i, \Gamma') | \exists \Gamma.\ (c, \Gamma) \in C \cup_\times C' \ \wedge\ \Gamma' \in \mathrm{branches}([\, \Gamma \,]^n_i)\} \\
&= \{([\, c_1 \cup c_2 \,]^{\Gamma_1 \cup \Gamma_2}_i, \Gamma') | \exists \Gamma_1 \Gamma_2.\ (c_1, \Gamma_1) \in C \ \wedge\ (c_2, \Gamma_2) \in C' \ \wedge\ \Gamma_1, \Gamma_2 \text{ are compatible} \wedge \\
&\qquad \Gamma' \in \mathrm{branches}([\, \Gamma_1 \cup \Gamma_2 \,]^n_i)\} \\
&= \{([\, c_1 \,]^{\Gamma_1}_i \cup [\, c_2 \,]^{\Gamma_2}_i, \Gamma') | \exists \Gamma_1 \Gamma_2.\ (c_1, \Gamma_1) \in C \ \wedge\ (c_2, \Gamma_2) \in C' \ \wedge\ \Gamma_1, \Gamma_2 \text{ are compatible} \wedge \\
&\qquad \Gamma' \in \mathrm{branches}([\, \Gamma_1 \,]^n_i \cup [\, \Gamma_2 \,]^n_i)\} \\
&= \{([\, c_1 \,]^{\Gamma_1}_i \cup [\, c_2 \,]^{\Gamma_2}_i, \Gamma \cup \Gamma') | \exists \Gamma_1 \Gamma_2.\ (c_1, \Gamma_1) \in C \ \wedge\ (c_2, \Gamma_2) \in C' \ \wedge\ \Gamma_1, \Gamma_2 \text{ are compatible} \wedge \\
&\qquad \Gamma \in \mathrm{branches}([\, \Gamma_1 \,]^n_i) \ \wedge\ \Gamma' \in \mathrm{branches}([\, \Gamma_2 \,]^n_i) \ \wedge\ \Gamma, \Gamma' \text{ are compatible}\}
\end{aligned}$$

The last step is proved by directly showing both inclusions.

On the other hand we have:

$$\begin{aligned}
[\, C \,]^n_i \cup_\times [\, C' \,]^n_i &= \{(c \cup c', \Gamma \cup \Gamma') | (c, \Gamma) \in [\, C \,]^n_i \ \wedge\ (c', \Gamma') \in [\, C' \,]^n_i \ \wedge\ \Gamma, \Gamma' \text{ are compatible}\} \\
&= \{([\, c_1 \,]^{\Gamma_1}_i \cup [\, c_2 \,]^{\Gamma_2}_i, \Gamma \cup \Gamma') | \exists \Gamma_1 \Gamma_2.\ (c_1, \Gamma_1) \in C \ \wedge\ (c_2, \Gamma_2) \in C' \wedge \\
&\qquad \Gamma \in \mathrm{branches}([\, \Gamma_1 \,]^n_i) \ \wedge\ \Gamma' \in \mathrm{branches}([\, \Gamma_2 \,]^n_i) \ \wedge\ \Gamma, \Gamma' \text{ are compatible}\} \\
&= \{([\, c_1 \,]^{\Gamma_1}_i \cup [\, c_2 \,]^{\Gamma_2}_i, \Gamma \cup \Gamma') | \exists \Gamma_1 \Gamma_2.\ (c_1, \Gamma_1) \in C \ \wedge\ (c_2, \Gamma_2) \in C' \wedge \\
&\qquad \Gamma \in \mathrm{branches}([\, \Gamma_1 \,]^n_i) \ \wedge\ \Gamma' \in \mathrm{branches}([\, \Gamma_2 \,]^n_i) \wedge \\
&\qquad \Gamma, \Gamma' \text{ are compatible} \ \wedge\ \Gamma_1, \Gamma_2 \text{ are compatible}\}
\end{aligned}$$

This last step comes from the fact that if $(c_1, \Gamma_1) \in C$ and $(c_2, \Gamma_2) \in C'$, then by assumption Γ_1 and Γ_2 do not contain union types. This implies that if $\Gamma \in \mathrm{branches}([\, \Gamma_1 \,]^n_i)$ and $\Gamma' \in \mathrm{branches}([\, \Gamma_2 \,]^n_i)$ are compatible, then Γ_1 and Γ_2 are compatible. Indeed, let $x \in \mathrm{dom}(\Gamma_1) \cap \mathrm{dom}(\Gamma_2)$. Hence $x_i \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma')$, and since they are compatible, $\Gamma(x_i) = \Gamma'(x_i)$. That is to say that there exists $T \in \mathrm{branches}([\, \Gamma_1(x) \,]^n) \cap \mathrm{branches}([\, \Gamma_2(x) \,]^n)$.

If $\Gamma_1(x) = [\![\tau^{l,\infty}_m \,;\, \tau^{l',\infty}_p]\!]$ (for some m, p, l, l'), then $[\, \Gamma_1(x) \,]^n = \bigvee_{1 \le j \le n} [\![\tau^{l,1}_{m_j} \,;\, \tau^{l',1}_{p_j}]\!]$, and thus there exists $j \in [\![1, n]\!]$ such that $T = [\![\tau^{l,1}_{m_j} \,;\, \tau^{l',1}_{p_j}]\!]$. Hence, $[\![\tau^{l,1}_{m_j} \,;\, \tau^{l',1}_{p_j}]\!] \in \mathrm{branches}([\, \Gamma_2(x) \,]^n)$. Because of the definition

of $[\,\cdot\,]^n$, and since $\Gamma_2(x)$ is not a union type (by assumption), this implies that $\Gamma_2(x) = [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]$, and therefore $\Gamma_1(x) = \Gamma_2(x)$.

If $\Gamma_1(x)$ is not of the form $[\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]$ (for some m, p, l, l'), then neither is $\Gamma_2(x)$ (by contraposition, following the same reasoning as in the previous case). $\Gamma_1(x)$ and $\Gamma_2(x)$ are not of the form $T' \vee T''$ either, by assumption. Therefore, neither $[\,\Gamma_1(x)\,]^n$ nor $[\,\Gamma_2(x)\,]^n$ are union types (from the definition of $[\,\cdot\,]^n$). This implies that $T = [\,\Gamma_1(x)\,]^n = [\,\Gamma_2(x)\,]^n$, which implies $\Gamma_1(x) = \Gamma_2(x)$.

In both cases $\Gamma_1(x) = \Gamma_2(x)$, and Γ_1, Γ_2 are therefore compatible.

Hence $[\,C\,]_i^n \cup_\times [\,C'\,]_i^n = [\,C \cup_\times C'\,]_i^n$, which proves the claim.

The second point directly follows from the definition of $[\,\cdot\,]_i^n$ and $\cup_\forall$. Indeed, for all C, c, Γ satisfying the assumptions, we have:

$$\begin{aligned}
[\,C \cup_\forall c\,]_i^n &= \{([\,c'\,]_i^{\Gamma'}, \Gamma'') | \exists \Gamma'.\ (c', \Gamma') \in C \cup_\forall c \ \wedge\ \Gamma'' \in \mathrm{branches}([\,\Gamma'\,]_i^n)\} \\
&= \{([\,c'' \cup c\,]_i^{\Gamma'}, \Gamma'') | \exists \Gamma'.\ (c'', \Gamma') \in C \ \wedge\ \Gamma'' \in \mathrm{branches}([\,\Gamma'\,]_i^n)\} \\
&= \{([\,c''\,]_i^{\Gamma'} \cup [\,c\,]_i^{\Gamma'}, \Gamma'') | \exists \Gamma'.\ (c'', \Gamma') \in C \ \wedge\ \Gamma'' \in \mathrm{branches}([\,\Gamma'\,]_i^n)\} \\
&= \{([\,c''\,]_i^{\Gamma'} \cup [\,c\,]_i^{\Gamma}, \Gamma'') | \exists \Gamma'.\ (c'', \Gamma') \in C \ \wedge\ \Gamma'' \in \mathrm{branches}([\,\Gamma'\,]_i^n)\} \\
&\qquad \text{(since } \Gamma, \Gamma' \text{ give the same types to names and keys)} \\
&= \{([\,c'\,]_i^{\Gamma'}, \Gamma'') | \exists \Gamma'.\ (c', \Gamma') \in C \ \wedge\ \Gamma'' \in \mathrm{branches}([\,\Gamma'\,]_i^n)\} \cup_\forall [\,c\,]_i^{\Gamma} \\
&= [\,C\,]_i^n \cup_\forall [\,c\,]_i^{\Gamma}
\end{aligned}$$

Theorem 6 (Typing processes with expanded types). *For all Γ, P, Q and C, if*

$$\Gamma \vdash P \sim Q \to C$$

then for all $i, n \in \mathbb{N}$ such that $1 \leq i \leq n$, there exists $C' \subseteq [\,C\,]_i^n$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C'$$

Proof. We prove this theorem by induction on the derivation Π of $\Gamma \vdash P \sim Q \to C$. We distinguish several cases for the last rule applied in this derivation.

- PZERO: then $P = Q = [\,P\,]_i^\Gamma = [\,Q\,]_i^\Gamma = \mathtt{0}$, and $C = \{(\emptyset, \Gamma)\}$. Hence

  $$[\,C\,]_i^n = \{(\emptyset, \Gamma') \mid \Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)\}$$

  Thus, by applying rule POR as many times as necessary to split $[\,\Gamma\,]_i^n$ into all of its branches, followed by rule PZERO, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to [\,C\,]_i^n$.
- POUT: then $P = \mathtt{out}(M).P'$, $Q = \mathtt{out}(N).Q'$ for some messages M, N and some processes P', Q', and

  $$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash P' \sim Q' \to C'} \qquad \dfrac{\Pi''}{\Gamma \vdash M \sim N : \mathtt{LL} \to c}}{\Gamma \vdash P \sim Q \to C = C' \cup_\forall c}.$$

  By applying the induction hypothesis to Π', there exists $C''' \subseteq [\,C'\,]_i^n$ and a proof Π''' of $[\,\Gamma\,]_i^n \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C'''$.

  Hence, by Lemma 9, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exist $C_{\Gamma'} \subseteq C'''$ and a proof $\Pi_{\Gamma'}$ of $\Gamma' \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C_{\Gamma'}$.

Moreover, by Lemma 29, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exists a proof $\Pi'_{\Gamma'}$ of $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \mathtt{LL} \to [\,c\,]_i^\Gamma$.
In addition, $[\,P\,]_i^\Gamma = [\,\mathtt{out}(M).P'\,]_i^\Gamma = \mathtt{out}([\,M\,]_i^\Gamma).[\,P'\,]_i^\Gamma$. Similarly, $[\,Q\,]_i^\Gamma = \mathtt{out}([\,N\,]_i^\Gamma).[\,Q'\,]_i^\Gamma$.
Therefore, using $\Pi_{\Gamma'}$, $\Pi'_{\Gamma'}$ and rule POUT, we have for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$ that $\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} \cup_\forall [\,c\,]_i^\Gamma \subseteq C'' \cup_\forall [\,c\,]_i^\Gamma$.
Thus by Lemma 32, there exists $C_1 \subseteq C'' \cup_\forall [\,c\,]_i^\Gamma$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1.$$

Finally, $[\,C\,]_i^n = [\,C' \cup_\forall c\,]_i^n = [\,C'\,]_i^n \cup_\forall [\,c\,]_i^\Gamma$ (by Lemma 33, whose conditions are satisfied, by Lemma 14).
Hence $C'' \cup_\forall [\,c\,]_i^\Gamma \subseteq [\,C\,]_i^n$, which proves the claim.

– PIN: then $P = \mathtt{in}(x).P'$, $Q = \mathtt{in}(x).Q'$ for some variable x and some processes P', Q', and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma, x : \mathtt{LL} \vdash P' \sim Q' \to C}}{\Gamma \vdash P \sim Q \to C}.$$

Since

$$[\,\Gamma, x : \mathtt{LL}\,]_i^n = [\,\Gamma\,]_i^n, x_i : [\,\mathtt{LL}\,]^n = [\,\Gamma\,]_i^n, x_i : \mathtt{LL},$$

by applying the induction hypothesis to Π', there exists $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of $[\,\Gamma\,]_i^n, x : \mathtt{LL} \vdash [\,P'\,]_i^{\Gamma, x:\mathtt{LL}} \sim [\,Q'\,]_i^{\Gamma, x:\mathtt{LL}} \to C'$.
In addition, $[\,P\,]_i^\Gamma = [\,\mathtt{in}(x).P'\,]_i^\Gamma = \mathtt{in}(x_i).[\,P'\,]_i^\Gamma = \mathtt{in}(x_i).[\,P'\,]_i^{\Gamma, x:\mathtt{LL}}$. Similarly, we have $[\,Q\,]_i^\Gamma = \mathtt{in}(x_i).[\,Q'\,]_i^{\Gamma, x:\mathtt{LL}}$.
Therefore, using Π'' and rule PIN, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C' \subseteq [\,C\,]_i^n$.

– PNEW: then $P = \mathtt{new}\ m : \tau_m^{l,a}.P'$, $Q = \mathtt{new}\ m : \tau_m^{l,a}.Q'$ for some m, l, a and some processes P', Q', and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma, m : \tau_m^{l,a} \vdash P' \sim Q' \to C}}{\Gamma \vdash P \sim Q \to C}.$$

- If $a = 1$:
  Since
  $$[\,\Gamma, m : \tau_m^{l,1}\,]_i^n = [\,\Gamma\,]_i^n, m : \tau_m^{l,1},$$
  by applying the induction hypothesis to Π', there exists $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of $[\,\Gamma\,]_i^n, m : \tau_m^{l,1} \vdash [\,P'\,]_i^{\Gamma, m:\tau_m^{l,1}} \sim [\,Q'\,]_i^{\Gamma, m:\tau_m^{l,1}} \to C'$.
  In addition $[\,P\,]_i^\Gamma = [\,\mathtt{new}\ m.P'\,]_i^\Gamma = \mathtt{new}\ m : \tau_m^{l,1}.[\,P'\,]_i^\Gamma = \mathtt{new}\ m : \tau_m^{l,1}.[\,P'\,]_i^{\Gamma, m:\tau_m^{l,1}}$; and similarly for Q. Therefore, using Π'' and rule PNEW, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C' \subseteq [\,C\,]_i^n$.
- If $a = \infty$: Since
  $$[\,\Gamma, m : \tau_m^{l,\infty}\,]_i^n = [\,\Gamma\,]_i^n, m_i : \tau_{m_i}^{l,1},$$
  by applying the induction hypothesis to Π', there exists $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of $[\,\Gamma\,]_i^n, m_i : \tau_{m_i}^{l,1} \vdash [\,P'\,]_i^{\Gamma, m:\tau_m^{l,\infty}} \sim [\,Q'\,]_i^{\Gamma, m:\tau_m^{l,\infty}} \to C'$.
  In addition $[\,P\,]_i^\Gamma = [\,\mathtt{new}\ m.P'\,]_i^\Gamma = \mathtt{new}\ m_i : \tau_{m_i}^{l,1}.[\,P'[m_i/m]\,]_i^\Gamma = \mathtt{new}\ m_i : \tau_{m_i}^{l,1}.[\,P'\,]_i^{\Gamma, m:\tau_m^{l,\infty}}$; and similarly for Q. Therefore, using Π'' and rule PNEW, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C' \subseteq [\,C\,]_i^n$.

– PNEWKEY: this case is similar to the PNEW case.
– PPAR: then $P = P' \mid P''$, $Q = Q' \mid Q''$ for some processes P', Q', P'', Q'', and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash P' \sim Q' \to C'} \qquad \dfrac{\Pi''}{\Gamma \vdash P'' \sim Q'' \to C''}}{\Gamma \vdash P \sim Q \to C = C' \cup_{\times} C''}.$$

By applying the induction hypothesis to Π', there exists $C''' \subseteq [\,C'\,]_i^n$ and a proof Π''' of $[\,\Gamma\,]_i^n \vdash [\,P'\,]_i^{\Gamma} \sim [\,Q'\,]_i^{\Gamma} \to C'''$. Similarly, by applying the induction hypothesis to Π'', there exists $C'''' \subseteq [\,C''\,]_i^n$ and a proof Π'''' of $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^{\Gamma} \sim [\,Q''\,]_i^{\Gamma} \to C''''$.
In addition, $[\,P\,]_i^{\Gamma} = [\,P' \mid P''\,]_i^{\Gamma} = [\,P'\,]_i^{\Gamma} \mid [\,P''\,]_i^{\Gamma}$. Similarly, $[\,Q\,]_i^{\Gamma} = [\,Q' \mid Q''\,]_i^{\Gamma} = [\,Q'\,]_i^{\Gamma} \mid [\,Q''\,]_i^{\Gamma}$.
Finally, $[\,C\,]_i^n = [\,C' \cup_{\times} C''\,]_i^n = [\,C'\,]_i^n \cup_{\times} [\,C''\,]_i^n$, by Lemma 33 (using Lemma 11 to ensure the condition that the environments do not contain union types).
Therefore, using Π''', Π'''' and rule PPAR, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^{\Gamma} \sim [\,Q\,]_i^{\Gamma} \to C''' \cup_{\times} C'''' \subseteq [\,C\,]_i^n$.
– POR: then $\Gamma = \Gamma', x : T \vee T'$ for some Γ', some $x \in \mathcal{X}$ and some types T, T', and

$$\Pi = \frac{\dfrac{\Pi_T}{\Gamma', x : T \vdash P \sim Q \to C'} \qquad \dfrac{\Pi_{T'}}{\Gamma', x : T' \vdash P \sim Q \to C''}}{\Gamma \vdash P \sim Q \to C = C' \cup C''}.$$

By applying the induction hypothesis to Π_T, there exist $C_1 \subseteq [\,C'\,]_i^n$ and a proof Π_1 of $[\,\Gamma'\,]_i^n, x_i : [\,T\,]^n \vdash [\,P\,]_i^{\Gamma', x:T} \sim [\,Q\,]_i^{\Gamma', x:T} \to C_1$. Similarly with $\Pi_{T'}$, there exist $C_2 \subseteq [\,C''\,]_i^n$ and a proof Π_2 of $[\,\Gamma'\,]_i^n, x_i : [\,T'\,]^n \vdash [\,P\,]_i^{\Gamma', x:T'} \sim [\,Q\,]_i^{\Gamma', x:T'} \to C_2$.
In addition $[\,P\,]_i^{\Gamma} = [\,P\,]_i^{\Gamma', x:T} = [\,P\,]_i^{\Gamma', x:T'}$, and similarly for Q.
Thus by rule POR, we have

$$[\,\Gamma'\,]_i^n, x_i : [\,T\,]^n \vee [\,T'\,]^n \vdash [\,P\,]_i^{\Gamma} \sim [\,Q\,]_i^{\Gamma} \to C_1 \cup C_2 \subseteq [\,C'\,]_i^n \cup [\,C''\,]_i^n = [\,C\,]_i^n.$$

Since $[\,\Gamma\,]_i^n = [\,\Gamma', x : T \vee T'\,]_i^n = [\,\Gamma'\,]_i^n, x_i : [\,T\,]^n \vee [\,T'\,]^n$, this proves the claim in this case.
– PLET: then $P = \texttt{let}\ x = t\ \texttt{in}\ P'\ \texttt{else}\ P''$, $Q = \texttt{let}\ x = t'\ \texttt{in}\ Q'\ \texttt{else}\ Q''$ for some variable x and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\dfrac{\Pi_d}{\Gamma \vdash_d t \sim t' : T} \qquad \dfrac{\Pi'}{\Gamma, x : T \vdash P' \sim Q' \to C'} \qquad \dfrac{\Pi''}{\Gamma \vdash P'' \sim Q'' \to C''}}{\Gamma \vdash P \sim Q \to C = C' \cup C''}.$$

Since

$$[\,\Gamma, x : T\,]_i^n = [\,\Gamma\,]_i^n, x_i : [\,T\,]^n,$$

by applying the induction hypothesis to Π', there exist $C''' \subseteq [\,C'\,]_i^n$ and a proof Π''' of $[\,\Gamma\,]_i^n, x_i : [\,T\,]^n \vdash [\,P'\,]_i^{\Gamma, x:T} \sim [\,Q'\,]_i^{\Gamma, x:T} \to C'''$. Similarly, there exist $C'''' \subseteq [\,C''\,]_i^n$ and a proof Π'''' of $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^{\Gamma} \sim [\,Q''\,]_i^{\Gamma} \to C''''$.
By Lemma 30 applied to Π_d, we also have

$$[\,\Gamma\,]_i^n \vdash_d [\,t\,]_i^{\Gamma} \sim [\,t'\,]_i^{\Gamma} : [\,T\,]^n$$

In addition, $[\,P\,]_i^{\Gamma} = [\,\texttt{let}\ x = t\ \texttt{in}\ P'\ \texttt{else}\ P''\,]_i^{\Gamma} = \texttt{let}\ x_i = [\,t\,]_i^{\Gamma}\ \texttt{in}\ [\,P'\,]_i^{\Gamma}\ \texttt{else}\ [\,P''\,]_i^{\Gamma} = \texttt{let}\ x_i = [\,t\,]_i^{\Gamma}\ \texttt{in}\ [\,P'\,]_i^{\Gamma, x:T}\ \texttt{else}\ [\,P''\,]_i^{\Gamma}$. Similarly, $[\,Q\,]_i^{\Gamma} = \texttt{let}\ x_i = [\,t'\,]_i^{\Gamma}\ \texttt{in}\ [\,Q'\,]_i^{\Gamma, x:T}\ \texttt{else}\ [\,Q''\,]_i^{\Gamma}$.
Therefore, using Π'' and rule PLET, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^{\Gamma} \sim [\,Q\,]_i^{\Gamma} \to C''' \cup C'''' \subseteq [\,C\,]_i^n$.

– PLETDEC: then $P = \texttt{let}\ x = \texttt{dec}(y, k_1)\ \texttt{in}\ P'\ \texttt{else}\ P''$, $Q = \texttt{let}\ x = \texttt{dec}(y, k_2)\ \texttt{in}\ Q'\ \texttt{else}\ Q''$ for some variable x, some keys k_1, k_2, and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\begin{array}{c}\Gamma(y) = \texttt{LL} \qquad \Gamma(k_1, k_2) <: \mathrm{key}^{\texttt{HH}}(T) \qquad \dfrac{\Pi'}{\Gamma, x : T \vdash P' \sim Q' \to C'} \qquad \dfrac{\Pi''}{\Gamma \vdash P'' \sim Q'' \to C''} \\ (\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_1, k_3) <: \mathrm{key}^{\texttt{HH}}(T') \Rightarrow \dfrac{\Pi^{1,k_3}}{\Gamma, x : T' \vdash P' \sim Q'' \to C_{k_3}}) \\ (\forall T'.\forall k_3 \neq k_2.\ \Gamma(k_3, k_2) <: \mathrm{key}^{\texttt{HH}}(T') \Rightarrow \dfrac{\Pi^{2,k_3}}{\Gamma, x : T' \vdash P'' \sim Q' \to C'_{k_3}})\end{array}}{\begin{array}{c}\Gamma \vdash \texttt{let}\ x = \texttt{dec}(y, k_1)\ \texttt{in}\ P'\ \texttt{else}\ P'' \sim \texttt{let}\ x = \texttt{dec}(y, k_2)\ \texttt{in}\ Q'\ \texttt{else}\ Q'' \to \\ C' \cup C'' \cup (\bigcup_{k_3} C_{k_3}) \cup (\bigcup_{k_3} {C'}_{k_3})\end{array}}$$

Let us write the proof for the case where $\Gamma(k_1, k_2) \neq \mathrm{seskey}^{\texttt{HH},\infty}(T)$. The other case is similar, although the keys are renamed, and slightly easier, since by well-formedness of Γ there are no k_3 satisfying the assumptions of the last two premises.
We thus have $[\, k_1 \,]_i^\Gamma = k_1$, $[\, k_2 \,]_i^\Gamma = k_2$, and for any k_3 satisfying the assumptions of either of the last two premises, $[\, k_3 \,]_i^\Gamma = k_3$.
Since

$$[\, \Gamma, x : T \,]_i^n = [\, \Gamma \,]_i^n, x_i : [\, T \,]^n,$$

by applying the induction hypothesis to Π', there exist $C'_1 \subseteq [\, C' \,]_i^n$ and a proof Π'_1 of

$$[\, \Gamma \,]_i^n, x_i : [\, T \,]^n \vdash [\, P' \,]_i^{\Gamma, x:T} \sim [\, Q' \,]_i^{\Gamma, x:T} \to C'_1.$$

Similarly, there exist $C''_1 \subseteq [\, C'' \,]_i^n$ and a proof Π''_1 of

$$[\, \Gamma \,]_i^n \vdash [\, P'' \,]_i^\Gamma \sim [\, Q'' \,]_i^\Gamma \to C''_1.$$

Similarly, for each $k_3 \neq k_2$ such that $\Gamma(k_1, k_3) = \mathrm{key}^{\texttt{HH}}(T')$ for some T', there exist $C_{k_3,1} \subseteq [\, C_{k_3} \,]_i^n$ and a proof Π_1^{1,k_3} of

$$[\, \Gamma \,]_i^n, x_i : [\, T' \,]^n \vdash [\, P' \,]_i^\Gamma \sim [\, Q'' \,]_i^\Gamma \to C_{k_3,1}.$$

Similarly, for each $k_3 \neq k_1$ such that $\Gamma(k_3, k_2) = \mathrm{key}^{\texttt{HH}}(T')$ for some T', there exist $C'_{k_3,1} \subseteq \left[\, C'_{k_3} \,\right]_i^n$ and a proof Π_1^{2,k_3} of

$$[\, \Gamma \,]_i^n, x_i : [\, T' \,]^n \vdash [\, P'' \,]_i^\Gamma \sim [\, Q' \,]_i^\Gamma \to C'_{k_3,1}.$$

Moreover, by definition, $[\, \Gamma \,]_i^n(y_i) = \texttt{LL}$, and for all l, T, and all keys k, k' that are either k_1, k_2, or a k_3 such as in the premises of the rule, if $\Gamma(k, k') <: \mathrm{key}^l(T)$ then $[\, \Gamma \,]_i^n(k, k') <: \mathrm{key}^l([\, T \,]^n)$.
In addition,

$$[\, P \,]_i^\Gamma = [\, \texttt{let}\ x = \texttt{dec}(y, k_1)\ \texttt{in}\ P'\ \texttt{else}\ P'' \,]_i^\Gamma = \texttt{let}\ x_i = \texttt{dec}(y_i, k_1)\ \texttt{in}\ [\, P' \,]_i^\Gamma\ \texttt{else}\ [\, P'' \,]_i^\Gamma,$$

i.e.

$$[\, P \,]_i^\Gamma = \texttt{let}\ x_i = \texttt{dec}(y_i, k_1)\ \texttt{in}\ [\, P' \,]_i^{\Gamma, x:T}\ \texttt{else}\ [\, P'' \,]_i^\Gamma.$$

Similarly, $[\, Q \,]_i^\Gamma = \texttt{let}\ x_i = \texttt{dec}(y_i, k_2)\ \texttt{in}\ [\, Q' \,]_i^{\Gamma, x:T}\ \texttt{else}\ [\, Q'' \,]_i^\Gamma$.
Therefore, using Π'_1, Π''_1, the Π_1^{1,k_3} and Π_1^{2,k_3}, and rule PLETDEC, we have $[\, \Gamma \,]_i^n \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C'_1 \cup C''_1 \cup (\bigcup_{k_3} C_{k_3,1}) \cup (\bigcup_{k_3} {C'}_{k_3,1}) \subseteq [\, C \,]_i^n$.

– PLETADECSAME, PLETADECDIFF: these cases are similar to the PLETDEC case.
– PLETLRK: then $P = \texttt{let}\ x = d(y)\ \texttt{in}\ P'\ \texttt{else}\ P''$, $Q = \texttt{let}\ x = d(y)\ \texttt{in}\ Q'\ \texttt{else}\ Q''$ for some variable $x \in \mathcal{X}$ and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\Gamma(y) = [\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!] \vee \Gamma(y) <: \mathrm{key}^l(T) \qquad \dfrac{\Pi'}{\Gamma \vdash P'' \sim Q'' \to C}}{\Gamma \vdash P \sim Q \to C}$$

for some m, p.
By applying the induction hypothesis to Π', there exists $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C'$.
We have $[\,P\,]_i^\Gamma = [\,\texttt{let}\ x = d(y)\ \texttt{in}\ P'\ \texttt{else}\ P''\,]_i^\Gamma = \texttt{let}\ x_i = d(y_i)\ \texttt{in}\ [\,P'\,]_i^\Gamma\ \texttt{else}\ [\,P''\,]_i^\Gamma$. Similarly, $[\,Q\,]_i^\Gamma = \texttt{let}\ x_i = d(y_i)\ \texttt{in}\ [\,Q'\,]_i^\Gamma\ \texttt{else}\ [\,Q''\,]_i^\Gamma$.
Let us first prove the case where $\Gamma(y) = [\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!]$.
We distinguish two cases, depending on whether the types in the refinement $[\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!]$ are finite nonce types or infinite nonce types, *i.e.* whether a is 1 or ∞.

- If a is 1: Then by definition of $[\,\Gamma\,]_i^n$, we have $[\,\Gamma\,]_i^n(y_i) = [\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!]$.
  Therefore, using Π'' and rule PLETLRK, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C' \subseteq [\,C\,]_i^n$.
- If a is ∞: Then by definition of $[\,\Gamma\,]_i^n$, we have $[\,\Gamma\,]_i^n(y_i) = \bigvee_{1 \le j \le n} [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$.
  Let $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$. By definition, there exists $j \in [\![1, n]\!]$, such that $\Gamma'(y_i) = [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$.
  Using Π'' and Lemma 9, there exist $C'_{\Gamma'} \subseteq [\,C\,]_i^n$ and a derivation $\Pi''_{\Gamma'}$ of $\Gamma' \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C'_{\Gamma'}$. Therefore, using rule PLETLRK, we have, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, $\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C'_{\Gamma'} \subseteq [\,C\,]_i^n$. Thus, by Lemma 32, we have

  $$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C'',$$

  where $C'' \subseteq [\,C\,]_i^n$, which proves the claim in this case.

We now prove the case where $\Gamma(y) <: \mathrm{key}^l(T)$. By Lemma 3, we thus have

$$\Gamma(y) \in \{\mathrm{seskey}^{l,\infty}(T), \mathrm{seskey}^{l,1}(T), \mathrm{eqkey}^l(T), \mathrm{key}^l(T)\}.$$

In all cases we have $\Gamma(y_i) <: \mathrm{key}^l([\,T\,]^n)$. Hence using Π'' and rule PLETLRK, we have $[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C''$.
– PIFL: then $P = \texttt{if}\ M = M'\ \texttt{then}\ P'\ \texttt{else}\ P''$, $Q = \texttt{if}\ N = N'\ \texttt{then}\ Q'\ \texttt{else}\ Q''$ for some messages M, N, M', N', and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\dfrac{\Pi''}{\Gamma \vdash P'' \sim Q'' \to C''} \qquad \dfrac{\Pi_1}{\Gamma \vdash M \sim N : \mathtt{LL} \to c} \qquad \dfrac{\Pi_2}{\Gamma \vdash M' \sim N' : \mathtt{LL} \to c'} \qquad \dfrac{\Pi'}{\Gamma \vdash P' \sim Q' \to C'}}{\Gamma \vdash P \sim Q \to C = (C' \cup C'') \cup_\forall (c \cup c')}.$$

By applying the induction hypothesis to Π', there exists $C''' \subseteq [\,C'\,]_i^n$ and a proof Π''' of $[\,\Gamma\,]_i^n \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C'''$. Similarly, there exists $C'''' \subseteq [\,C''\,]_i^n$ and a proof Π'''' of $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C''''$.

Hence, by Lemma 9, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exist $C_{\Gamma'} \subseteq C'''$ and a proof $\Pi_{1,\Gamma'}$ of $\Gamma' \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C_{\Gamma'}$; as well as $C'_{\Gamma'} \subseteq C''''$ and a proof $\Pi_{2,\Gamma'}$ of $\Gamma' \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C'_{\Gamma'}$.
Moreover, by Lemma 29, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \mathtt{LL} \to [\,c\,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\,M'\,]_i^\Gamma \sim [\,N'\,]_i^\Gamma : \mathtt{LL} \to [\,c'\,]_i^\Gamma$.
In addition,

$$[\,P\,]_i^\Gamma = [\,\mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''\,]_i^\Gamma = \mathtt{if}\ [\,M\,]_i^\Gamma = [\,M'\,]_i^\Gamma\ \mathtt{then}\ [\,P'\,]_i^\Gamma\ \mathtt{else}\ [\,P''\,]_i^\Gamma.$$

Similarly, $[\,Q\,]_i^\Gamma = \mathtt{if}\ [\,N\,]_i^\Gamma = [\,N'\,]_i^\Gamma\ \mathtt{then}\ [\,Q'\,]_i^\Gamma\ \mathtt{else}\ [\,Q''\,]_i^\Gamma$.
Therefore, using $\Pi_{1,\Gamma'}$, $\Pi_{2,\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi'_{2,\Gamma'}$ and rule PIFL, we have

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to (C_{\Gamma'} \cup C'_{\Gamma'}) \cup_\forall ([\,c\,]_i^\Gamma \cup [\,c'\,]_i^\Gamma) \subseteq (C''' \cup C'''') \cup_\forall ([\,c\,]_i^\Gamma \cup [\,c'\,]_i^\Gamma).$$

Thus by Lemma 32, there exists $C_1 \subseteq (C''' \cup C'''') \cup_\forall ([\,c\,]_i^\Gamma \cup [\,c'\,]_i^\Gamma)$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1.$$

Finally, $[\,C\,]_i^n = [\,(C' \cup C'') \cup_\forall (c \cup c')\,]_i^n = ([\,C'\,]_i^n \cup [\,C''\,]_i^n) \cup_\forall ([\,c\,]_i^\Gamma \cup [\,c'\,]_i^\Gamma)$ (by Lemma 33, whose conditions are satisfied, by Lemma 14). Hence $(C''' \cup C'''') \cup_\forall ([\,c\,]_i^\Gamma \cup [\,c'\,]_i^\Gamma) \subseteq [\,C\,]_i^n$, which proves the claim.

– PIFP: then $P = \mathtt{if}\ M = t\ \mathtt{then}\ P'\ \mathtt{else}\ P''$, $Q = \mathtt{if}\ N = t\ \mathtt{then}\ Q'\ \mathtt{else}\ Q''$ for some messages M, N, some $t \in \mathcal{K} \cup \mathcal{N} \cup \mathcal{C}$, and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash P' \sim Q' \to C'} \quad \dfrac{\Pi''}{\Gamma \vdash P'' \sim Q'' \to C''} \quad \dfrac{\Pi_1}{\Gamma \vdash M \sim N : \mathtt{LL} \to c} \quad \dfrac{\Pi_2}{\Gamma \vdash t \sim t : \mathtt{LL} \to c'}}{\Gamma \vdash P \sim Q \to C = C' \cup C''}.$$

By applying the induction hypothesis to Π', there exist $C''' \subseteq [\,C'\,]_i^n$ and a proof Π''' of the judgement $[\,\Gamma\,]_i^n \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C'''$. Similarly, there exist $C'''' \subseteq [\,C''\,]_i^n$ and a proof Π'''' of $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C''''$.
Hence, by Lemma 9, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exist $C_{\Gamma'} \subseteq C'''$ and a proof $\Pi_{1,\Gamma'}$ of the judgement $\Gamma' \vdash [\,P'\,]_i^\Gamma \sim [\,Q'\,]_i^\Gamma \to C_{\Gamma'}$; as well as $C'_{\Gamma'} \subseteq C''''$ and a proof $\Pi_{2,\Gamma'}$ of $\Gamma' \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C'_{\Gamma'}$.
Moreover, by Lemma 29, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of the judgement $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : \mathtt{LL} \to [\,c\,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\,t\,]_i^\Gamma \sim [\,t\,]_i^\Gamma : \mathtt{LL} \to [\,c'\,]_i^\Gamma$.
Since $t \in \mathcal{K} \cup \mathcal{N} \cup \mathcal{C}$, we also have $[\,t\,]_i^\Gamma \in \mathcal{K} \cup \mathcal{N} \cup \mathcal{C}$.
In addition,

$$[\,P\,]_i^\Gamma = [\,\mathtt{if}\ M = M'\ \mathtt{then}\ P'\ \mathtt{else}\ P''\,]_i^\Gamma = \mathtt{if}\ [\,M\,]_i^\Gamma = [\,M'\,]_i^\Gamma\ \mathtt{then}\ [\,P'\,]_i^\Gamma\ \mathtt{else}\ [\,P''\,]_i^\Gamma.$$

Similarly, $[\,Q\,]_i^\Gamma = \mathtt{if}\ [\,N\,]_i^\Gamma = [\,N'\,]_i^\Gamma\ \mathtt{then}\ [\,Q'\,]_i^\Gamma\ \mathtt{else}\ [\,Q''\,]_i^\Gamma$.
Therefore, using $\Pi_{1,\Gamma'}$, $\Pi_{2,\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi'_{2,\Gamma'}$ and rule PIFP, we have for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} \cup C'_{\Gamma'} \subseteq C''' \cup C''''.$$

Thus by Lemma 32, there exists $C_1 \subseteq (C''' \cup C'''')$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1.$$

Finally, $[\,C\,]_i^n = [\,C' \cup C''\,]_i^n = [\,C'\,]_i^n \cup [\,C''\,]_i^n$ (by Lemma 33). Hence $(C''' \cup C'''') \subseteq [\,C\,]_i^n$, which proves the claim.

– PIFLR: then $P = \texttt{if } M_1 = M_2 \texttt{ then } P_\top \texttt{ else } P_\bot$, $Q = \texttt{if } N_1 = N_2 \texttt{ then } Q_\top \texttt{ else } Q_\bot$ for some messages M_1, N_1, M_2, N_2, and some processes $P_\top$, $Q_\top$, $P_\bot$, $Q_\bot$, and there exist m, p, m', p' such that

$$\Pi = \frac{\dfrac{\Pi_1}{\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,1} ; \tau_p^{l',1}]\!] \to c} \quad b = (\tau_m^{l,1} \stackrel{?}{=} \tau_{m'}^{l'',1}) \quad b' = (\tau_p^{l'',1} \stackrel{?}{=} \tau_{p'}^{l''',1}) \quad \dfrac{\dfrac{\Pi_2}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',1} ; \tau_{p'}^{l''',1}]\!] \to c'} \quad \Pi'}{\Gamma \vdash P_b \sim Q_{b'} \to C}}{\Gamma \vdash P \sim Q \to C}.$$

By applying the induction hypothesis to Π', there exist $C' \subseteq [\, C \,]_i^n$ and a proof Π'' of the judgement $[\, \Gamma \,]_i^n \vdash [\, P_b \,]_i^\Gamma \sim [\, Q_{b'} \,]_i^\Gamma \to C'$.
Hence, by Lemma 9, for all $\Gamma' \in \text{branches}([\, \Gamma \,]_i^n)$, there exist $C_{\Gamma'} \subseteq C'$, and a proof $\Pi_{\Gamma'}$ of $\Gamma' \vdash [\, P_b \,]_i^\Gamma \sim [\, Q_{b'} \,]_i^\Gamma \to C_{\Gamma'}$.
Moreover, by Lemma 29 applied to Π_1, for all $\Gamma' \in \text{branches}([\, \Gamma \,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of the judgement $\Gamma' \vdash [\, M_1 \,]_i^\Gamma \sim [\, N_1 \,]_i^\Gamma : [\![\tau_m^{l,1} ; \tau_p^{l',1}]\!] \to [\, c \,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\, M_2 \,]_i^\Gamma \sim [\, N_2 \,]_i^\Gamma : [\![\tau_{m'}^{l'',1} ; \tau_{p'}^{l''',1}]\!] \to [\, c' \,]_i^\Gamma$.
In addition,

$$[\, P \,]_i^\Gamma = [\, \texttt{if } M_1 = M_2 \texttt{ then } P_\top \texttt{ else } P_\bot \,]_i^\Gamma = \texttt{if } [\, M_1 \,]_i^\Gamma = [\, M_2 \,]_i^\Gamma \texttt{ then } [\, P_\top \,]_i^\Gamma \texttt{ else } [\, P_\bot \,]_i^\Gamma.$$

Similarly, $[\, Q \,]_i^\Gamma = \texttt{if } [\, N_1 \,]_i^\Gamma = [\, N_2 \,]_i^\Gamma \texttt{ then } [\, Q_\top \,]_i^\Gamma \texttt{ else } [\, Q_\bot \,]_i^\Gamma$.
Therefore, using $\Pi_{\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi'_{2,\Gamma'}$ and rule PIFLR, we have for all $\Gamma' \in \text{branches}([\, \Gamma \,]_i^n)$

$$\Gamma' \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C_{\Gamma'} \subseteq C' \subseteq [\, C \,]_i^n.$$

Thus by Lemma 32, there exists $C_1 \subseteq [\, C \,]_i^n$ such that

$$[\, \Gamma \,]_i^n \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C_1,$$

which proves the claim.
– PIFS: then $P = \texttt{if } M = M' \texttt{ then } P' \texttt{ else } P''$, $Q = \texttt{if } N = N' \texttt{ then } Q' \texttt{ else } Q''$ for some messages M, N, M', N', and some processes P', Q', P'', Q'', and

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash P'' \sim Q'' \to C} \quad \dfrac{\Pi_1}{\Gamma \vdash M \sim N : \texttt{LL} \to c} \quad \dfrac{\Pi_2}{\Gamma \vdash M' \sim N' : \texttt{HH} \to c'}}{\Gamma \vdash P \sim Q \to C}.$$

By applying the induction hypothesis to Π', there exists $C' \subseteq [\, C \,]_i^n$ and a proof Π'' of the judgement $[\, \Gamma \,]_i^n \vdash [\, P'' \,]_i^\Gamma \sim [\, Q'' \,]_i^\Gamma \to C'$.
Hence, by Lemma 9, for all $\Gamma' \in \text{branches}([\, \Gamma \,]_i^n)$, there exist $C_{\Gamma'} \subseteq C'$ and a proof $\Pi_{\Gamma'}$ of the judgement $\Gamma' \vdash [\, P' \,]_i^\Gamma \sim [\, Q' \,]_i^\Gamma \to C_{\Gamma'}$.
Moreover, by Lemma 29, for all $\Gamma' \in \text{branches}([\, \Gamma \,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of the judgement $\Gamma' \vdash [\, M \,]_i^\Gamma \sim [\, N \,]_i^\Gamma : \texttt{LL} \to [\, c \,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\, M' \,]_i^\Gamma \sim [\, N' \,]_i^\Gamma : \texttt{HH} \to [\, c' \,]_i^\Gamma$.
In addition,

$$[\, P \,]_i^\Gamma = [\, \texttt{if } M = M' \texttt{ then } P' \texttt{ else } P'' \,]_i^\Gamma = \texttt{if } [\, M \,]_i^\Gamma = [\, M' \,]_i^\Gamma \texttt{ then } [\, P' \,]_i^\Gamma \texttt{ else } [\, P'' \,]_i^\Gamma.$$

Similarly, $[\,Q\,]_i^\Gamma = \texttt{if}\ [\,N\,]_i^\Gamma = [\,N'\,]_i^\Gamma\ \texttt{then}\ [\,Q'\,]_i^\Gamma\ \texttt{else}\ [\,Q''\,]_i^\Gamma$.
Therefore, using $\Pi_{\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi'_{2,\Gamma'}$ and rule PIFS, we have for all $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} \subseteq C' \subseteq [\,C\,]_i^n.$$

Thus by Lemma 32, there exists $C_1 \subseteq [\,C\,]_i^n$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1,$$

which proves the claim.

– PIFI: then $P = \texttt{if}\ M = M'\ \texttt{then}\ P'\ \texttt{else}\ P''$, $Q = \texttt{if}\ N = N'\ \texttt{then}\ Q'\ \texttt{else}\ Q''$ for some messages M, N, M', N', and some processes P', Q', P'', Q'', and there exist types T, T', and names m, p, such that

$$\Pi = \frac{\dfrac{\Pi'}{\Gamma \vdash P'' \sim Q'' \to C} \qquad \dfrac{\Pi_1}{\Gamma \vdash M \sim N : T * T' \to c} \qquad \dfrac{\Pi_2}{\Gamma \vdash M' \sim N' : [\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!] \to c'}}{\Gamma \vdash P \sim Q \to C}.$$

By applying the induction hypothesis to Π', there exist $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of the judgement $[\,\Gamma\,]_i^n \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C'$.
Let $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$. By applying Lemma 9 to Π'', there exists $C_{\Gamma'} \subseteq C'$, such that there exists a proof $\Pi_{\Gamma'}$ of the judgement $\Gamma' \vdash [\,P''\,]_i^\Gamma \sim [\,Q''\,]_i^\Gamma \to C_{\Gamma'}$.
Moreover, by Lemma 29 applied to the proof Π_1, there exists a proof $\Pi'_{1,\Gamma'}$ of the type judgement $\Gamma' \vdash [\,M\,]_i^\Gamma \sim [\,N\,]_i^\Gamma : [\,T\,]^n * [\,T'\,]^n \to [\,c\,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of the judgement $\Gamma' \vdash [\,M'\,]_i^\Gamma \sim [\,N'\,]_i^\Gamma : \left[\,[\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!]\,\right]^n) \to [\,c'\,]_i^\Gamma$.
In addition,

$$[\,P\,]_i^\Gamma = [\,\texttt{if}\ M = M'\ \texttt{then}\ P'\ \texttt{else}\ P''\,]_i^\Gamma = \texttt{if}\ [\,M\,]_i^\Gamma = [\,M'\,]_i^\Gamma\ \texttt{then}\ [\,P'\,]_i^\Gamma\ \texttt{else}\ [\,P''\,]_i^\Gamma.$$

Similarly, $[\,Q\,]_i^\Gamma = \texttt{if}\ [\,N\,]_i^\Gamma = [\,N'\,]_i^\Gamma\ \texttt{then}\ [\,Q'\,]_i^\Gamma\ \texttt{else}\ [\,Q''\,]_i^\Gamma$.
We distinguish two cases.

- If a is 1: Then $\left[\,[\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!]\,\right]^n = [\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!]$, and using $\Pi_{\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi'_{2,\Gamma'}$ and rule PIFI, we have for all $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$

  $$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} \subseteq C' \subseteq [\,C\,]_i^n.$$

  Thus by Lemma 32, there exists $C_1 \subseteq [\,C\,]_i^n$ such that

  $$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1,$$

  which proves the claim in this case.
- If a is ∞: Moreover, by applying Lemma 7 to $\Pi'_{2,\Gamma'}$, there exists a type

  $$T'' \in \text{branches}(\left[\,[\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]\,\right]^n),$$

  such that there exists a proof $\Pi''_{2,\Gamma'}$ of $\Gamma' \vdash [\,M'\,]_i^\Gamma \sim [\,N'\,]_i^\Gamma : T'' \to [\,c'\,]_i^\Gamma$.

By definition, $\left[\, [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!] \,\right]^n = \bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$. Therefore, by definition of branches, there exists j such that $T'' = [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$.
Hence, using $\Pi_{\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi''_{2,\Gamma'}$, by applying rule PIFI, we have for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$ that

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} \subseteq C' \subseteq [\,C\,]_i^n.$$

Thus by Lemma 32, there exists $C_1 \subseteq [\,C\,]_i^n$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_1$$

which proves the claim in this case.

– PIFLR*: then $P = \mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P_\top\ \mathtt{else}\ P_\bot$, $Q = \mathtt{if}\ N_1 = N_2\ \mathtt{then}\ Q_\top\ \mathtt{else}\ Q_\bot$ for some messages M_1, N_1, M_2, N_2, and some processes $P_\top$, $Q_\top$, $P_\bot$, $Q_\bot$, and there exist m, p, l, l' such that

$$\Pi = \dfrac{\dfrac{\Pi_2}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_m^{l,\infty}\,;\,\tau_{p'}^{l',\infty}]\!] \to \emptyset} \qquad \dfrac{\Pi_\top}{\Gamma \vdash P_\top \sim Q_\top \to C_1} \qquad \dfrac{\Pi_\bot}{\Gamma \vdash P_\bot \sim Q_\bot \to C_2}}{\Gamma \vdash P \sim Q \to C = C_1 \cup C_2}\ \dfrac{\Pi_1}{\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!] \to \emptyset}.$$

By applying the induction hypothesis to $\Pi_\top$, there exist $C' \subseteq [\,C_1\,]_i^n \subseteq [\,C\,]_i^n$ and a proof Π' of $[\,\Gamma\,]_i^n \vdash [\,P_\top\,]_i^\Gamma \sim [\,Q_\top\,]_i^\Gamma \to C'$. Similarly with $\Pi_\bot$, there exist $C'' \subseteq [\,C_2\,]_i^n \subseteq [\,C\,]_i^n$ and a proof Π'' of $[\,\Gamma\,]_i^n \vdash [\,P_\bot\,]_i^\Gamma \sim [\,Q_\bot\,]_i^\Gamma \to C''$.
Hence, by Lemma 9, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exist constraint sets $C_{1,\Gamma'} \subseteq C' (\subseteq [\,C\,]_i^n)$, and $C_{2,\Gamma'} \subseteq C'' (\subseteq [\,C\,]_i^n)$, and proofs $\Pi_{\top,\Gamma'}$ and $\Pi_{\bot,\Gamma'}$ of $\Gamma' \vdash [\,P_\top\,]_i^\Gamma \sim [\,Q_\top\,]_i^\Gamma \to C_{1,\Gamma'}$ and $\Gamma' \vdash [\,P_\bot\,]_i^\Gamma \sim [\,Q_\bot\,]_i^\Gamma \to C_{2,\Gamma'}$.
Moreover, by Lemma 29 applied to Π_1, for all $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of $\Gamma' \vdash [\,M_1\,]_i^\Gamma \sim [\,N_1\,]_i^\Gamma : \bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!] \to [\,c\,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\,M_2\,]_i^\Gamma \sim [\,N_2\,]_i^\Gamma : \bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!] \to [\,c'\,]_i^\Gamma$.
Let $\Gamma' \in \mathrm{branches}([\,\Gamma\,]_i^n)$. By Lemma 7, there exists $T \in \mathrm{branches}(\bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!])$ such that there exists a proof $\Pi''_{1,\Gamma'}$ of $\Gamma' \vdash [\,M_1\,]_i^\Gamma \sim [\,N_1\,]_i^\Gamma : T \to [\,c\,]_i^\Gamma$.
Similarly, there exists $T' \in \mathrm{branches}(\bigvee_{1\le j\le n}[\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!])$ such that there exists a proof $\Pi''_{2,\Gamma'}$ of $\Gamma' \vdash [\,M_2\,]_i^\Gamma \sim [\,N_2\,]_i^\Gamma : T' \to [\,c\,]_i^\Gamma$.
By definition of branches, there exist j, j' such that $T = [\![\tau_{m_j}^{l,1}\,;\,\tau_{p_j}^{l',1}]\!]$ and $T' = [\![\tau_{m_{j'}}^{l,1}\,;\,\tau_{p_{j'}}^{l',1}]\!]$.
In addition,

$$[\,P\,]_i^\Gamma = [\,\mathtt{if}\ M_1 = M_2\ \mathtt{then}\ P_\top\ \mathtt{else}\ P_\bot\,]_i^\Gamma = \mathtt{if}\ [\,M_1\,]_i^\Gamma = [\,M_2\,]_i^\Gamma\ \mathtt{then}\ [\,P_\top\,]_i^\Gamma\ \mathtt{else}\ [\,P_\bot\,]_i^\Gamma.$$

Similarly, $[\,Q\,]_i^\Gamma = \mathtt{if}\ [\,N_1\,]_i^\Gamma = [\,N_2\,]_i^\Gamma\ \mathtt{then}\ [\,Q_\top\,]_i^\Gamma\ \mathtt{else}\ [\,Q_\bot\,]_i^\Gamma$.
Therefore, using $\Pi_{\top,\Gamma'}$, $\Pi_{\bot,\Gamma'}$, $\Pi''_{1,\Gamma'}$, $\Pi''_{2,\Gamma'}$ and rule PIFLR, either $j = j'$ and we have

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{1,\Gamma'} (\subseteq [\,C\,]_i^n)$$

or $j \neq j'$ and we have

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{2,\Gamma'} (\subseteq [\,C\,]_i^n).$$

This holds for any $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$.
Thus by Lemma 32, there exists $C' \subseteq [\,C\,]_i^n$ such that

$$[\,\Gamma\,]_i^n \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C'$$

which proves the claim in this case.

– PIFLR'*: then $P = \texttt{if}\ M_1 = M_2\ \texttt{then}\ P_\top\ \texttt{else}\ P_\bot$, $Q = \texttt{if}\ N_1 = N_2\ \texttt{then}\ Q_\top\ \texttt{else}\ Q_\bot$ for some messages M_1, N_1, M_2, N_2, and some processes $P_\top, Q_\top, P_\bot, Q_\bot$, and there exist names m, p, m', p' such that

$$\Pi = \dfrac{\dfrac{\dfrac{\Pi_1}{\Gamma \vdash M_1 \sim N_1 : [\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!] \to \emptyset}\quad \dfrac{\Pi_2}{\Gamma \vdash M_2 \sim N_2 : [\![\tau_{m'}^{l'',a'}\,;\,\tau_{p'}^{l''',a'}]\!] \to \emptyset} \qquad \dfrac{\Pi'}{\Gamma \vdash P_\bot \sim Q_\bot \to C}}{\Gamma \vdash P \sim Q \to C}.$$

By applying the induction hypothesis to Π', there exist $C' \subseteq [\,C\,]_i^n$ and a proof Π'' of the judgement $[\,\Gamma\,]_i^n \vdash [\,P_\bot\,]_i^\Gamma \sim [\,Q_\bot\,]_i^\Gamma \to C'$.
Hence, by Lemma 9, for all $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$, there exist $C_{\Gamma'} \subseteq C' (\subseteq [\,C\,]_i^n)$, and a proof $\Pi_{\Gamma'}$ of $\Gamma' \vdash [\,P_\bot\,]_i^\Gamma \sim [\,Q_\bot\,]_i^\Gamma \to C_{\Gamma'}$.
Moreover, by Lemma 29 applied to Π_1, for all $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$, there exists a proof $\Pi'_{1,\Gamma'}$ of $\Gamma' \vdash [\,M_1\,]_i^\Gamma \sim [\,N_1\,]_i^\Gamma : \left[\,[\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!]\,\right]^n \to [\,c\,]_i^\Gamma$. Similarly, there exists a proof $\Pi'_{2,\Gamma'}$ of $\Gamma' \vdash [\,M_2\,]_i^\Gamma \sim [\,N_2\,]_i^\Gamma : \left[\,[\![\tau_{m'}^{l'',a'}\,;\,\tau_{p'}^{l''',a'}]\!]\,\right]^n \to [\,c'\,]_i^\Gamma$.
We distinguish several cases, depending on a and a'.

- if a and a' are both 1: Then this rule is a particular case of rule PIFLR, and the result is proved in a similar way.
- if a is 1 and a' is ∞: Then $\left[\,[\![\tau_m^{l,a}\,;\,\tau_p^{l',a}]\!]\,\right]^n = \left[\,[\![\tau_m^{l,1}\,;\,\tau_p^{l',1}]\!]\,\right]^n$, and

$$\left[\,[\![\tau_{m'}^{l'',a}\,;\,\tau_{p'}^{l''',a}]\!]\,\right]^n = \bigvee_{1\le j\le n} [\![\tau_{m'_j}^{l'',1}\,;\,\tau_{p_j}^{l''',1}]\!].$$

  Let $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$.
  By Lemma 7, using $\Pi'_{2,\Gamma'}$, there exists $j \in [\![1,n]\!]$ such that there exists a proof $\Pi''_{2,\Gamma'}$ of $\Gamma' \vdash [\,M_2\,]_i^\Gamma \sim [\,N_2\,]_i^\Gamma : [\![\tau_{m'_j}^{l'',1}\,;\,\tau_{p'_j}^{l''',1}]\!] \to [\,c'\,]_i^\Gamma$.
  In addition,

$$[\,P\,]_i^\Gamma = [\,\texttt{if}\ M_1 = M_2\ \texttt{then}\ P_\top\ \texttt{else}\ P_\bot\,]_i^\Gamma = \texttt{if}\ [\,M_1\,]_i^\Gamma = [\,M_2\,]_i^\Gamma\ \texttt{then}\ [\,P_\top\,]_i^\Gamma\ \texttt{else}\ [\,P_\bot\,]_i^\Gamma.$$

  Similarly, $[\,Q\,]_i^\Gamma = \texttt{if}\ [\,N_1\,]_i^\Gamma = [\,N_2\,]_i^\Gamma\ \texttt{then}\ [\,Q_\top\,]_i^\Gamma\ \texttt{else}\ [\,Q_\bot\,]_i^\Gamma$.
  For any $j \in [\![1,n]\!]$, $\tau_m^{l,1} \neq \tau_{m'_j}^{l'',1}$; and $\tau_p^{l',1} \neq \tau_{p'_j}^{l''',1}$.
  Therefore, using $\Pi_{\Gamma'}$, $\Pi'_{1,\Gamma'}$, $\Pi''_{2,\Gamma'}$ and rule PIFLR, we have

$$\Gamma' \vdash [\,P\,]_i^\Gamma \sim [\,Q\,]_i^\Gamma \to C_{\Gamma'} (\subseteq [\,C\,]_i^n).$$

  This holds for any $\Gamma' \in \text{branches}([\,\Gamma\,]_i^n)$.

Thus by Lemma 32, there exists $C' \subseteq [\, C \,]_i^n$ such that

$$[\, \Gamma \,]_i^n \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C'$$

which proves the claim in this case.

- if a is ∞ and a' is 1: This case is similar to the symmetric one.
- if a and a' both are ∞: This case is similar to the case where a is 1 and a' is ∞.

– PIFALL: this case is similar to the PIFL case.

The next theorem corresponds to the first step mentioned in Subsection 6.4.

Theorem 7 (Typing n sessions). *For all Γ, P, Q and C, such that*

$$\Gamma \vdash P \sim Q \to C$$

then for all $n \geq 1$, there exists $C' \subseteq \cup_{\times 1 \leq i \leq n} [\, C \,]_i^n$ such that

$$[\, \Gamma \,]^n \vdash [\, P \,]_1^\Gamma \mid \ldots \mid [\, P \,]_n^\Gamma \sim [\, Q \,]_1^\Gamma \mid \ldots \mid [\, Q \,]_n^\Gamma \to C'$$

where $[\, \Gamma \,]^n$ is defined as $\bigcup_{1 \leq i \leq n} [\, \Gamma \,]_i^n$.

Proof. Let us assume Γ, P, Q and C are such that

$$\Gamma \vdash P \sim Q \to C.$$

The claim clearly holds (using Theorem 6) if $n = 1$. Let then $n \geq 2$.

Note that the union $\bigcup_{1 \leq i \leq n} [\, \Gamma \,]_i^n$ is well-defined, as for $i \neq j$, $\mathrm{dom}([\, \Gamma \,]_i^n) \cap \mathrm{dom}([\, \Gamma \,]_j^n) \subseteq \mathcal{K} \cup \mathcal{N}$, and the types associated to keys and nonces are the same in each $[\, \Gamma \,]_i^n$.

The property follows from Theorem 6. Indeed, this theorem guarantees that for all $i \in [\![1, n]\!]$, there exists $C_i \subseteq [\, C \,]_i^n$ such that

$$[\, \Gamma \,]_i^n \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C_i.$$

By construction, all variables in $\mathrm{dom}([\, \Gamma \,]_i^n)$ are indexed with i, and all keys and nonces are either unindexed or indexed with i, and as we mentioned earlier, for all i, j, $[\, \Gamma \,]_i^n$ and $[\, \Gamma \,]_j^n$ have the same values on their common domain.

Hence we have $[\, \Gamma \,]^n = [\, \Gamma \,]_i^n \uplus (\bigcup_{j \neq i} ([\, \Gamma \,]_j^n)|_{\mathrm{dom}([\, \Gamma \,]_j^n) \setminus \mathrm{dom}([\, \Gamma \,]_i^n)})$. Therefore, by Lemma 12, we have for all $i \in [\![1, n]\!]$

$$[\, \Gamma \,]^n \vdash [\, P \,]_i^\Gamma \sim [\, Q \,]_i^\Gamma \to C_i'$$

where

$$C_i' = \{(c, \Gamma' \cup \Gamma'') | (c, \Gamma') \in C_i \ \wedge \ \Gamma'' \in \mathrm{branches}(\bigcup_{j \neq i} ([\, \Gamma \,]_j^n)|_{d_{i,j}})\}$$

and

$$d_{i,j} = \mathrm{dom}([\, \Gamma \,]_j^n) \setminus \mathrm{dom}([\, \Gamma \,]_i^n).$$

Thus, by applying rule PPAR $n - 1$ times, we have

$$[\, \Gamma \,]^n \vdash [\, P \,]_1^\Gamma \mid \ldots \mid [\, P \,]_n^\Gamma \sim [\, Q \,]_1^\Gamma \mid \ldots [\, Q \,]_n^\Gamma \to \cup_{\times 1 \leq i \leq n} C_i'.$$

It only remains to be proved that $\cup_{\times 1 \leq i \leq n} C_i' \subseteq \cup_{\times 1 \leq i \leq n} [\, C \,]_i^n$. Since for all $i \in [\![1, n]\!]$ we have $C_i \subseteq [\, C \,]_i^n$, by Lemma 13 we know that $\cup_{\times 1 \leq i \leq n} C_i \subseteq \cup_{\times 1 \leq i \leq n} [\, C \,]_i^n$.

Hence it suffices to show that $\cup_{\times 1\leq i\leq n} C'_i \subseteq \cup_{\times 1\leq i\leq n} C_i$.

Let $(c, \Gamma') \in \cup_{\times 1\leq i\leq n} C'_i$. By definition there exist $(c_1, \Gamma_1) \in C'_1, \ldots, (c_n, \Gamma_n) \in C'_n$ such that $c = \bigcup_{1\leq i\leq n} c_i$, $\Gamma' = \bigcup_{1\leq i\leq n} \Gamma_i$, and for all $i \neq j$, Γ_i and Γ_j are compatible.

For all i, $(c_i, \Gamma_i) \in C'_i$. Thus by definition of C'_i there exist Γ'_i and Γ''_i such that $(c_i, \Gamma'_i) \in C_i$, $\Gamma''_i \in \mathrm{branches}(\bigcup_{j\neq i} ([\,\Gamma\,]^n_j)|_{d_{i,j}})$, and $\Gamma_i = \Gamma'_i \cup \Gamma''_i$. Since for all $i \neq j$, Γ_i and Γ_j are compatible, we know that Γ'_i and Γ'_j also are, as well as Γ'_i and Γ''_j.

Hence, $\Gamma' = \bigcup_{1\leq i\leq n} \Gamma_i = \bigcup_{1\leq i\leq n} (\Gamma'_i \cup \Gamma''_i) = (\bigcup_{1\leq i\leq n} \Gamma'_i) \cup (\bigcup_{1\leq i\leq n} \Gamma''_i)$.

Moreover, $(\bigcup_{1\leq i\leq n} \Gamma''_i) = (\bigcup_{1\leq i\leq n} \Gamma'_i)$. Indeed, they have the same domain, *i.e.*

$$\bigcup_{i\neq j} d_{i,j} = \bigcup_{i\neq j} \mathrm{dom}([\,\Gamma\,]^n_j) \backslash \mathrm{dom}([\,\Gamma\,]^n_i) = \bigcup_i \mathrm{dom}([\,\Gamma\,]^n_i)$$

since $n \geq 2$, and are compatible since for all $i \neq j$, Γ'_i and Γ''_j are compatible.

Thus $\Gamma' = (\bigcup_{1\leq i\leq n} \Gamma'_i)$, and since the Γ'_i are all pairwise compatible, and for all i, $(c_i, \Gamma'_i) \in C_i$, we have $(c, \Gamma') \in \cup_{\times 1\leq i\leq n} C_i$.

This proves that $\cup_{\times 1\leq i\leq n} C'_i \subseteq \cup_{\times 1\leq i\leq n} C_i$, which concludes the proof.

This next theorem, together with Theorem 11, entails Theorem 3:

Theorem 8. *Consider P, Q, P' ,Q', C, C', such that P, Q and P', Q' do not share any variable. Consider Γ, containing only keys and nonces with types of the form $\tau^{l,1}_n$.*

Assume that P and Q only bind nonces and keys with infinite nonce types, i.e. *using* `new` $m : \tau^{l,\infty}_m$ *and* `new` $k : \mathrm{seskey}^{l,\infty}(T)$ *for some label l and type T; while P' and Q' only bind nonces and keys with finite types,* i.e. *using* `new` $m : \tau^{l,1}_m$ *and* `new` $k : \mathrm{seskey}^{l,1}(T)$.

Let us abbreviate by `new` $\overline{n}$ *the sequence of declarations of each nonce $m \in \mathrm{dom}(\Gamma)$ and session key k such that $\Gamma(k, k) = \mathrm{seskey}^{l,1}(T)$ for some l, T. If*

- $\Gamma \vdash P \sim Q \to C$,
- $\Gamma \vdash P' \sim Q' \to C'$,
- $C' \cup_\times (\cup_{\times 1\leq i\leq n} [\,C\,]^n_i)$ *is consistent for all $n \geq 1$,*

then

$$\texttt{new}\ \overline{n}.\ ((!P) \mid P') \approx_t \texttt{new}\ \overline{n}.\ ((!Q) \mid Q').$$

Proof. Note that since Γ only contains keys and nonces with finite types, for all i, $[\,P\,]^\Gamma_i = [\,P\,]^\emptyset_i$ is just P where all variables and some names have been α-renamed, and similarly for Q. Since P', Q' only contain nonces with finite types, $[\,P'\,]^\Gamma_1$ and $[\,Q'\,]^\Gamma_1$ are P', Q' where all variables have been α-renamed.

By Theorem 7, we know that for all i, n,

$$[\,\Gamma\,]^n \vdash [\,P\,]^\Gamma_1 \mid \ldots \mid [\,P\,]^\Gamma_n \sim [\,Q\,]^\Gamma_1 \mid \ldots \mid [\,Q\,]^\Gamma_n \to C''$$

where $[\,\Gamma\,]^n = \bigcup_{1\leq i\leq n} [\,\Gamma\,]^n_i$, and $C'' \subseteq \cup_{\times 1\leq i\leq n} [\,C\,]^n_i$.

By Theorem 6, there also exists $C''' \subseteq [\,C'\,]^n_1$, such that

$$[\,\Gamma\,]^n_1 \vdash [\,P'\,]^\Gamma_1 \sim [\,Q'\,]^\Gamma_1 \to C'''.$$

Therefore, by Lemma 12, we have

$$[\,\Gamma\,]^n \vdash [\,P'\,]^\Gamma_1 \sim [\,Q'\,]^\Gamma_1 \to C''''$$

where C'''' is C''' where all the environments have been extended with $\bigcup_{1\leq i\leq n}([\,\Gamma\,]_i^n)_{\mathcal{N},\mathcal{K}}$ (note that this environment still only contains nonces and keys).

Therefore, by rules PPAR and PNEW,

$$\Gamma' \vdash \mathtt{new}\ \overline{n}.\ ([\,P\,]_1^\Gamma \mid \ldots \mid [\,P\,]_n^\Gamma) \mid [\,P'\,]_1^\Gamma \sim \mathtt{new}\ \overline{n}.\ ([\,Q\,]_1^\Gamma \mid \ldots \mid [\,Q\,]_n^\Gamma) \mid [\,Q'\,]_1^\Gamma \to C''\cup_\times C''''$$

where Γ' is the restriction of $[\,\Gamma\,]^n$ to keys.

If $[\,C'\,]_1^n \cup_\times (\cup_{\times\, 1\leq i\leq n}[\,C\,]_i^n)$ is consistent, similarly to the reasoning in the proof of Theorem 7, $C''\cup_\times C''''$ also is.

Then, by Theorem 5,

$$\mathtt{new}\ \overline{n}.\ ([\,P\,]_1^\Gamma \mid \ldots \mid [\,P\,]_n^\Gamma) \mid [\,P'\,]_1^\Gamma \approx_t \mathtt{new}\ \overline{n}.\ ([\,Q\,]_1^\Gamma \mid \ldots \mid [\,Q\,]_n^\Gamma) \mid [\,Q'\,]_1^\Gamma$$

which implies (since $[\,P'\,]_1^\Gamma$ is just a renaming of the variables in P') that

$$\mathtt{new}\ \overline{n}.\ ([\,P\,]_1^\Gamma \mid \ldots \mid [\,P\,]_n^\Gamma) \mid P' \approx_t \mathtt{new}\ \overline{n}.\ ([\,Q\,]_1^\Gamma \mid \ldots \mid [\,Q\,]_n^\Gamma) \mid Q'$$

Since $[\,P\,]_i^\Gamma$ and $[\,Q\,]_i^\Gamma$ are just α-renamings of P, Q, this implies that for all n,

$$\mathtt{new}\ \overline{n}.\ (P_1 \mid \ldots \mid P_n) \mid P' \approx_t \mathtt{new}\ \overline{n}.\ (Q_1 \mid \ldots \mid Q_n) \mid Q'$$

where $P_1 = \cdots = P_n = P$, and $Q_1 = \cdots = Q_n = Q$. Therefore

$$\mathtt{new}\ \overline{n}.\ ((!P) \mid P') \approx_t \mathtt{new}\ \overline{n}.\ ((!Q) \mid Q').$$

B.3 Checking consistency

In this subsection, we first recall in detail the `check_const` procedure presented in Section 6.3, which was described in Section 6.3,and prove its correctness in the non-replicated case.

For a constraint c and an environment Γ, let

$$\mathtt{step1}_\Gamma(c) := ([\![c]\!]_{\sigma_F,\sigma'_F}, \Gamma'),$$

where

$$F = \{x \in \mathrm{dom}(\Gamma) \mid \exists m,n,l,l'.\ \Gamma(x) = [\![\tau_m^{l,1}\,;\,\tau_n^{l',1}]\!]\},$$

σ_F, σ'_F are the substitutions defined by

- $\mathrm{dom}(\sigma_F) = \mathrm{dom}(\sigma'_F) = F$
- $\forall x \in F.\ \forall m,n,l,l'.\Gamma(x) = [\![\tau_m^{l,1}\,;\,\tau_n^{l',1}]\!] \Rightarrow \sigma_F(x) = m\ \wedge\ \sigma'_F(x) = n,$

and Γ' is the environment obtained by extending the restriction of Γ to $\mathrm{dom}(\Gamma)\backslash F$ with $\Gamma'(n) = \tau_n^{l,1}$ for all nonce n such that $\tau_n^{l,1}$ occurs in Γ. This is well defined, since by assumption on the well-formedness of the processes and by definition of the processes, a name n is always associated with the same label.

We define the condition $\mathtt{step2}_\Gamma(c)$ as: check that c only contains elements of the form $M \sim N$ where M and N are both

- $\mathtt{enc}(M', M'')$, $\mathtt{enc}(N', N'')$ where M'', N'' are either

  - keys k, k' where $\exists T.\Gamma(k,k') <: \text{key}^{\mathtt{HH}}(T)$;
  - or a variable x such that $\exists T.\Gamma(x) <: \text{key}^{\mathtt{HH}}(T)$;
- or encryptions $\mathtt{aenc}(M', M'')$, $\mathtt{aenc}(N', N'')$ where
  - M' and N' contain directly under pairs a nonce n such that $\Gamma(n) = \tau_n^{\mathtt{HH},a}$ or a secret key k such that $\exists T, k'.\Gamma(k,k') <: \text{key}^{\mathtt{HH}}(T)$ or $\Gamma(k',k) <: \text{key}^{\mathtt{HH}}(T)$, or a variable x such that $\exists m,n,a.\Gamma(x) = [\![\tau_m^{\mathtt{HH},a}\,;\,\tau_n^{\mathtt{HH},a}]\!]$, or a variable x such that $\exists T.\Gamma(x) <: \text{key}^{\mathtt{HH}}(T)$;
  - M'' and N'' are either
    - public keys $\mathtt{pk}(k)$, $\mathtt{pk}(k')$ where $\exists T.\Gamma(k,k') <: \text{key}^{\mathtt{HH}}(T)$;
    - or public keys $\mathtt{pk}(x)$, $\mathtt{pk}(x)$ where $\exists T.\Gamma(x) <: \text{key}^{\mathtt{HH}}(T)$;
    - or a variable x such that $\exists T, T'.\Gamma(x) = \text{pkey}(T)$ and $T <: \text{key}^{\mathtt{HH}}(T')$;
- or hashes $\mathtt{h}(M')$, $\mathtt{h}(N')$, where M', N' similarly contain a secret value under pairs;
- or signatures $\mathtt{sign}(M', M'')$, $\mathtt{sign}(N', M'')$ where M'', N'' are either
  - keys k, k' where $\exists T.\Gamma(k,k') <: \text{key}^{\mathtt{HH}}(T)$;
  - or a variable x such that $\exists T.\Gamma(x) <: \text{key}^{\mathtt{HH}}(T)$;

$\mathtt{step2}_\Gamma(c)$ returns $\mathtt{true}$ if this check succeeds and $\mathtt{false}$ otherwise.

We then proceed to $\mathtt{step3}$. We define condition $\mathtt{step3}_\Gamma(c)$ as follows. We consider all $M \sim M' \in c$ and $N \sim N' \in c$, such that M, N are unifiable with a most general unifier μ, and such that

$$\forall x \in \text{dom}(\mu).\ \forall l, l', m, p.\ (\Gamma(x) = [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]) \Rightarrow (x\mu \in \mathcal{X}\ \vee\ \exists i.\ x\mu = m_i).$$

We then define the substitution θ, over all variables $x \in \text{dom}(\mu)$ such that $\Gamma(x) = [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]$ by

$$\forall x \in \text{dom}(\mu).\ \forall l, l', m, p, i. \quad (\Gamma(x) = [\![\tau_m^{l,\infty}\,;\,\tau_p^{l',\infty}]\!]\ \wedge\ \mu(x) = m_i) \Rightarrow \theta(x) = p_i$$

and $\theta(x) = x$ otherwise.

Let then α be the restriction of μ to $\{x \in \text{dom}(\mu) \mid \Gamma(x) = \mathtt{LL}\ \wedge\ \mu(x) \in \mathcal{N}\}$.

We then check that $M'\alpha\theta = N'\alpha\theta$.

Similarly, we check that the symmetric condition, when M' and N' are unifiable, holds for all $M \sim M' \in c$ and $N \sim N' \in c$.

If all these checks succeed, $\mathtt{step3}_\Gamma(c)$ returns $\mathtt{true}$.

Finally, $\mathtt{check\_const}(C)$ is computed by considering all $(c, \Gamma) \in C$. We let $(\overline{c}, \overline{\Gamma}) = \mathtt{step1}_\Gamma(c)$. We then check that $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$ and $\mathtt{step3}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$. If this check succeeds for all $(c, \Gamma) \in C$, we say that $\mathtt{check\_const}(C) = \mathtt{true}$.

Note that we only consider constraints obtained by typing, and therefore such that Γ is well-formed, and such that there exists $c_\phi \subseteq c$ such that $\Gamma \vdash \phi_\ell(c) \sim \phi_r(c) : \mathtt{LL} \to c_\phi$.

Indeed, it is clear by induction on the typing rules for terms that:

$$\forall \Gamma, M, N, T, c. \quad \Gamma \vdash M \sim N : T \to c \Longrightarrow (\forall u \sim v \in c. \quad \exists c' \subseteq c. \quad \Gamma \vdash u \sim v : \mathtt{LL} \to c').$$

From this result, and using Lemmas 5 and 14, it follows clearly by induction on the typing rules for processes that

$$\forall \Gamma, P, Q, C.\quad \Gamma \vdash P \sim Q \to C \Longrightarrow (\forall (c, \Gamma') \in C.\quad \forall u \sim v \in c.\quad \exists c' \subseteq c.\quad \Gamma' \vdash u \sim v : \mathtt{LL} \to c').$$

Let us now prove that the procedure is correct for constraints without infinite nonce types, *i.e.* constraint sets C such that

$$\forall (c, \Gamma) \in C.\, \forall l, l', m, n.\, \Gamma(x) \neq [\![\tau_m^{l,\infty}\,;\, \tau_n^{l',\infty}]\!].$$

We fix such a constraint set C (obtained by typing).

Let $(c, \Gamma) \in C$. Let $(\overline{c}, \overline{\Gamma}) = \mathtt{step1}_{\Gamma}(c)$. Let us assume that $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$ and $\mathtt{step3}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$.

Lemma 34. *If $\overline{c}$ is consistent in $\overline{\Gamma}$, then c is consistent in Γ.*

Proof. Let c' be a set of constraints and Γ' be a typing environment such that $c' \subseteq c$, $\Gamma' \subseteq \Gamma$, ${\Gamma'}_{\mathcal{N},\mathcal{K}} = \Gamma_{\mathcal{N},\mathcal{K}}$ and $\mathrm{vars}(c') \subseteq \mathrm{dom}(\Gamma')$. Let σ, σ' be two substitutions such that ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c_\sigma$ (for some set of constraints $c_\sigma \subseteq [\![c']\!]_{\sigma,\sigma'}$).

To prove the claim, we need to show that the frames $\phi_{\mathrm{LL}}^{\Gamma} \cup \phi_\ell([\![c']\!]_{\sigma,\sigma'}))$ and $(\phi_{\mathrm{LL}}^{\Gamma} \cup \phi_r([\![c']\!]_{\sigma,\sigma'}))$ are statically equivalent. Let D denote $\mathrm{dom}({\Gamma'}_{\mathcal{X}})(= \mathrm{dom}(\sigma) = \mathrm{dom}(\sigma'))$.

For all $x \in F \cap D$, by definition of F, there exist m, n, l, l' such that $\Gamma(x) = [\![\tau_m^{l,1}\,;\, \tau_n^{l',1}]\!]$. Thus, by well-typedness of σ, σ', there exists c_x such that $\Gamma \vdash \sigma(x) \sim \sigma'(x) : [\![\tau_m^{l,1}\,;\, \tau_n^{l',1}]\!] \to c_x$. Hence, by Lemma 16, since σ, σ' are ground, we have $\sigma(x) = m$ and $\sigma'(x) = n$. Therefore, $\sigma|_{D\cap F} = \sigma_F|_D$ and $\sigma'|_{D\cap F} = \sigma'_F|_D$.

Let c'' be the set $[\![c']\!]_{\sigma|_{D\cap F},\sigma'|_{D\cap F}}$. By Lemma 13, we have $[\![c']\!]_{\sigma,\sigma'} = [\![c'']\!]_{\sigma|_{D\setminus F},\sigma'|_{D\setminus F}}$. We also have $c'' = [\![c']\!]_{\sigma_F|_D,\sigma'_F|_D}$, which is equal to $[\![c']\!]_{\sigma_F,\sigma'_F}$ since $\mathrm{vars}(c') \subseteq D$. Hence $c'' \subseteq \overline{c}$.

Let $\Gamma'' = \Gamma'|_{\mathrm{dom}(\Gamma')\setminus F}$. We have $\Gamma'' \subseteq \overline{\Gamma}$.

Moreover, since ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c_\sigma$, it is clear from the definition of well-typedness for substitutions that we also have ${\Gamma''}_{\mathcal{N},\mathcal{K}} \vdash \sigma|_{D\setminus F} \sim \sigma'|_{D\setminus F} : {\Gamma''}_{\mathcal{X}} \to c'_\sigma$ for some $c'_\sigma \subseteq c_\sigma$. Note that $c'_\sigma \subseteq c_\sigma \subseteq [\![c']\!]_{\sigma,\sigma'} = [\![c'']\!]_{\sigma|_{D\setminus F},\sigma'|_{D\setminus F}}$. Finally, $\mathrm{vars}(c'') \subseteq \mathrm{vars}(c')\setminus F$ by definition of instantiation, thus $\mathrm{vars}(c'') \subseteq \mathrm{dom}(\Gamma'')$.

We have established that $c'' \subseteq \overline{c}$, $\Gamma'' \subseteq \overline{\Gamma}$, $\mathrm{vars}(c'') \subseteq \mathrm{dom}(\Gamma'')$, ${\Gamma''}_{\mathcal{N},\mathcal{K}} \vdash \sigma|_{D\setminus F} \sim \sigma'|_{D\setminus F} : {\Gamma''}_{\mathcal{X}} \to c'_\sigma$, and $c'_\sigma \subseteq [\![c'']\!]_{\sigma|_{D\setminus F},\sigma'|_{D\setminus F}}$. Therefore, by definition of the consistency of $\overline{c}$ in $\overline{\Gamma}$, the frames $\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![c'']\!]_{\sigma|_{D\setminus F},\sigma'|_{D\setminus F}})$ and $\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![c'']\!]_{\sigma|_{D\setminus F},\sigma'|_{D\setminus F}})$ are statically equivalent.

Since $\phi_{\mathrm{LL}}^{\overline{\Gamma}} \subseteq \phi_{\mathrm{LL}}^{\Gamma}$, that is to say that $\phi_{\mathrm{LL}}^{\Gamma} \cup \phi_\ell([\![c']\!]_{\sigma,\sigma'})$ and $\phi_{\mathrm{LL}}^{\Gamma} \cup \phi_r([\![c']\!]_{\sigma,\sigma'})$ are statically equivalent. This proves the consistency of c in Γ.

Lemma 35. *For all ground σ, σ' such that $\exists \Gamma' \subseteq \overline{\Gamma}.\, \exists c_\sigma.\, {\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c_\sigma$, we have*

$$\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}.$$

Proof. Let $M \sim N \in [\![\overline{c}]\!]_{\sigma,\sigma'}$. By definition there exist $M' \sim N' \in \overline{c}$ such that $M = M'\sigma$ and $N = N'\sigma'$. Since $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$, there are four cases for M' and N': they can be symmetric or asymmetric encryptions, hashes, or signatures. These four cases are similar, we write the proof for the asymmetric encryption case.

In this case, there exist M'_1, M'_2, N'_1, N'_2 such that $M' = \mathtt{aenc}(M'_1, M'_2)$ and $N' = \mathtt{aenc}(N'_1, N'_2)$. Let us show that $M'\sigma \sim N'\sigma'$ satisfies the conditions for $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'})$.

Since $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$ we know that M'_1 and N'_1 contain directly under pairs

– a nonce n such that $\Gamma(n) = \tau_n^{\mathtt{HH},a}$
– or a secret key k such that $\exists T, k'.\Gamma(k,k') <: \mathrm{key}^{\mathtt{HH}}(T)$ or $\Gamma(k',k) <: \mathrm{key}^{\mathtt{HH}}(T)$,
– or a variable x such that $\exists m,n,a.\Gamma(x) = [\![\tau_m^{\mathtt{HH},a}\,;\,\tau_n^{\mathtt{HH},a}]\!]$,
– or a variable x such that $\exists T.\Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$.

In the fist two cases, $M_1'\sigma$ (resp. $N_1'\sigma'$) clearly contains the same nonce or keys under pairs. The last two cases are similar, we write the proof for the case of a variable x such that $\overline{\Gamma}(x) = [\![\tau_m^{\mathtt{HH},a}\,;\,\tau_n^{\mathtt{HH},a}]\!]$. Either $x \notin \mathrm{dom}(\sigma)$, and thus x still appears under pairs in $M_1'\sigma$; or $x \in \mathrm{dom}(\sigma)$. In that case, $\sigma(x)$ appear directly under pairs in $M_1'\sigma$ (resp. $\sigma'(x)$ in $N_1'\sigma'$). By assumption, there exists c' such that $\overline{\Gamma} \vdash \sigma(x) \sim \sigma'(x) : [\![\tau_m^{\mathtt{HH},a}\,;\,\tau_n^{\mathtt{HH},a}]\!] \to c'$. Hence, by Lemma 16, $\sigma(x) = m$ (resp. $\sigma'(x) = n$) and $\overline{\Gamma}(n) = \tau_n^{\mathtt{HH},a}$ (resp. $\overline{\Gamma}(m) = \tau_m^{\mathtt{HH},a}$). Therefore, in all cases, $M_1'\sigma$ and $N_1'\sigma'$ satisfy the required conditions.

In addition, since $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$ we also know that M_2' and N_2' are either

– public keys $\mathtt{pk}(k)$, $\mathtt{pk}(k')$ where $\exists T.\Gamma(k,k') <: \mathrm{key}^{\mathtt{HH}}(T)$;
– or public keys $\mathtt{pk}(x)$, $\mathtt{pk}(x)$ where $\exists T.\Gamma(x) <: \mathrm{key}^{\mathtt{HH}}(T)$;
– or a variable x such that $\exists T, T'.\Gamma(x) = \mathrm{pkey}(T)$ and $T <: \mathrm{key}^{\mathtt{HH}}(T')$;

In the first case $M_2'\sigma = M_2'$ and $N_2'\sigma' = N_2'$ are still the same public keys. In the two other cases, similarly to the proof for M_1' and N_1', either $x \notin \mathrm{dom}(\sigma)$, and we also have $M_2'\sigma = M_2'$ and $N_2'\sigma' = N_2'$; or $x \in \mathrm{dom}(\sigma)$, and by well-typedness of σ, σ' and Lemma 20, $M_2'\sigma$ and $N_2'\sigma'$ are keys of the right type. Therefore, in all cases, $M_2'\sigma$ and $N_2'\sigma'$ satisfy the required conditions.

Thus, $M'\sigma \sim N'\sigma'$ satisfies the conditions for $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'})$.

Lemma 36. *There exists $c_\phi \subseteq \overline{c}$ such that $\overline{\Gamma} \vdash \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell(\overline{c}) \sim \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r(\overline{c}) : \mathtt{LL} \to c_\phi$.*

Proof. As explained previously, there exists $c'_\phi \subseteq c$ such that $\Gamma \vdash \phi_\ell(c) \sim \phi_r(c) : \mathtt{LL} \to c'_\phi$.

Moreover, we have by definition $\Gamma = \Gamma_F \uplus \overline{\Gamma}'$, where F is defined as in $\mathtt{step1}$ and Γ_F is the restriction of Γ to F, and for some $\overline{\Gamma}' \subseteq \overline{\Gamma}$. In addition $(\Gamma_F)_{\mathcal{N},\mathcal{K}} \vdash \sigma_F \sim \sigma'_F : (\Gamma_F)_{\mathcal{X}} \to \emptyset$ using rule TLR[1]. By definition of F, and since the refinement types in Γ only contain ground terms by assumption, we also know that $\overline{\Gamma}$ does not contain refinement types. Hence, by Lemma 24, and Lemma 12, there exists $c_\phi \subseteq [\![c]\!]_{\sigma_F,\sigma'_F}$, *i.e.* $c_\phi \subseteq \overline{c}$ such that $\overline{\Gamma} \vdash \phi_\ell(c)\sigma_F \sim \phi_r(c)\sigma'_F : \mathtt{LL} \to c_\phi$. Since $\overline{c} = [\![c]\!]_{\sigma_F,\sigma'_F}$, this proves that $\overline{\Gamma} \vdash \phi_\ell(\overline{c}) \sim \phi_r(\overline{c}) : \mathtt{LL} \to c_\phi$. Besides, it is clear from the definition of $\phi_{\mathtt{LL}}^{\Gamma}$ and rules TCstFN, TNonceL, TKey, TPubKeyL, TVKeyL that $\overline{\Gamma} \vdash \phi_{\mathtt{LL}}^{\Gamma} \sim \phi_{\mathtt{LL}}^{\Gamma} : \mathtt{LL} \to \emptyset$.

These two results prove the lemma.

Lemma 37. *For all ground σ, σ' such that $\exists \Gamma' \subseteq \overline{\Gamma}.\ \exists c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}.\ \Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c_\sigma$, there exists $c_\phi \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$ such that $\overline{\Gamma} \vdash \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}) \sim \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}) : \mathtt{LL} \to c_\phi$.*

Proof. By Lemma 36, there exists $c'_\phi \subseteq \overline{c}$ such that $\overline{\Gamma} \vdash \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell(\overline{c}) \sim \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r(\overline{c}) : \mathtt{LL} \to c'_\phi$. Since $\Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c_\sigma$, and since $(\overline{\Gamma} \backslash \Gamma'_{\mathcal{X}})$ does not contain variables with finite nonce types, by Lemma 24, there exists $c_\phi \subseteq \left[\!\!\left[c'_\phi\right]\!\!\right]_{\sigma,\sigma'} \cup c_\sigma$ such that $\overline{\Gamma} \backslash \Gamma'_{\mathcal{X}} \vdash (\phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell(\overline{c}))\sigma \sim (\phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r(\overline{c}))\sigma' : \mathtt{LL} \to c_\phi$.

Hence, by Lemma 12, we have $\overline{\Gamma} \vdash \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_\ell(\overline{c})\sigma \sim \phi_{\mathtt{LL}}^{\Gamma} \cup \phi_r(\overline{c})\sigma' : \mathtt{LL} \to c_\phi$.

In addition, $c'_\phi \subseteq \overline{c}$ and $c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$, we have $\left[\!\!\left[c'_\phi\right]\!\!\right]_{\sigma,\sigma'} \cup c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$. Therefore $c_\phi \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$, which concludes the proof.

Lemma 38. *For all ground σ, σ', for all recipe R such that*

- $\exists \Gamma' \subseteq \overline{\Gamma}.\ \exists c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}.\ \Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c_\sigma$,
- $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell(\overline{c}))$,
- $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$ *and* $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$,
- $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})$ *and* $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$ *restricted to* $\mathrm{vars}(R)$ *are ground,*

there exists a recipe R' *without destructors,* i.e. *in which* `dec`, `adec`, `checksign`, π_1, π_2, *do not appear, such that*

- $\mathrm{vars}(R') \subseteq \mathrm{vars}(R)$,
- $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$,
- $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$.

Proof. We prove the property by induction on R.

- If $R = x \in \mathcal{AX}$ or $R = a \in \mathcal{C} \cup \mathcal{FN}$ then the claim holds with $R' = R$.
- If the head symbol of R is a constructor,*i.e.* if there exist R_1, R_2 such that $R = \mathtt{pk}(R_1)$ or $R = \mathtt{vk}(R_1)$ or $R = \mathtt{enc}(R_1, R_2)$ or $R = \mathtt{aenc}(R_1, R_2)$ or $R = \mathtt{sign}(R_1, R_2)$ or $R = \langle R_1, R_2 \rangle$ or $R = \mathtt{h}(R_1)$, we may apply the induction hypothesis to R_1 (and R_2 when it is present). All these case are similar, we write the proof generically for $R = f(R_1, R_2)$. By the induction hypothesis, there exist R'_1, R'_2 such that
  - for all $i \in \{1, 2\}$, $\mathrm{vars}(R'_i) \subseteq \mathrm{vars}(R_i)$,
  - for all $i \in \{1, 2\}$, $R_i(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'_i(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$,
  - for all $i \in \{1, 2\}$, $R_i(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'_i(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$.

  Let $R' = f(R'_1, R'_2)$. The first point imply that R' satisfies the conditions on variables. Since $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = f(R_1(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow, R_2(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow)$, the second point implies that $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$. Similarly, $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$, and the claim holds.
- If $R = \mathtt{dec}(S, K)$ for some recipes S, K, then since $R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$, we have

$$K(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = k$$

  for some $k \in K$, and $S(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{enc}(M, k)$, where $M = R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$.
  Similarly, there exists $k' \in \mathcal{K}$ such that $K(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = k'$ and $S(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{enc}(N, k')$, where $N = R(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$.
  In addition, by Lemma 37, there exists c' such that $\overline{\Gamma} \vdash \phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}) \sim \phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}) : \mathtt{LL} \to c'$. Thus by Lemma 23, there exists c'' such that $\overline{\Gamma} \vdash K(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \sim K(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow : \mathtt{LL} \to c''$, which is to say $\overline{\Gamma} \vdash k \sim k' : \mathtt{LL} \to c''$. Hence by Lemma 20 and by well-formedness of Γ, $k = k'$ and $\Gamma(k, k) <: \mathrm{key}^{\mathtt{LL}}(T)$ for some type T.
  Since $S(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{enc}(M, k) \neq \bot$, and $\mathrm{vars}(S) \subseteq \mathrm{vars}(R)$, by the induction hypothesis, there exists S' such that $\mathrm{vars}(S') \subseteq \mathrm{vars}(S)$, $S'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{enc}(M, k)$, and $S'(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi^{\overline{\Gamma}}_{\mathtt{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{enc}(N, k)$.
  It is then clear that either $S' = x$ for some variable $x \in \mathcal{AX}$, or $S' = \mathtt{enc}(S'', K')$ for some S'', K'. We have already shown that $\Gamma(k, k) <: \mathrm{key}^{\mathtt{LL}}(T)$. In addition, by Lemma 35, $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$. Hence $[\![\overline{c}]\!]_{\sigma,\sigma'}$ only contains messages encrypted with keys k'' such that $\Gamma(k'', k''') <: \mathrm{key}^{\mathtt{HH}}(T')$ or $\Gamma(k''', k'') <: \mathrm{key}^{\mathtt{HH}}(T')$ for some T', k'''. Therefore the first case is not possible.

Hence there exist S'', K' such that $S' = \mathtt{enc}(S'', K')$. Since $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \mathtt{enc}(M, k)$, we have $S''(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = M$. Hence $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = M = S''(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, and similarly for $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$. Moreover, S'' being a subterm of S' itd also satisfies the conditions on the domains, and thus the property holds with $R' = S''$.

– If $R = \mathtt{adec}(S, K)$ for some recipes S, K: this case is similar to the symmetric case.

– If $R = \mathtt{checksign}(S, K)$ for some recipes S, K: then since $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$, we have $K(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{vk}(k)$ for some $k \in K$, and $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{sign}(M, k)$, where $M = R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$.

Similarly, there exists $k' \in \mathcal{K}$ such that $K(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{vk}(k')$ and $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{sign}(N, k')$, where $N = R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$.

Since $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{sign}(M, k) \neq \bot$, by the induction hypothesis, there exists S' such that $\mathrm{vars}(S') \subseteq \mathrm{vars}(S)$, $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{sign}(M, k)$, and $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \mathtt{sign}(N, k)$.

Since $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \mathtt{sign}(M, k)$, it is clear from the definition of $\downarrow$ that either $S' = x$ for some $x \in \mathcal{AX}$, or $S' = \mathtt{sign}(S'', K')$ for some S'', K'.

- In the first case, we therefore have $\mathtt{sign}(M, k) \sim \mathtt{sign}(N, k') \in [\![\overline{c}]\!]_{\sigma,\sigma'}$. In addition, by Lemma 37, there exists $c' \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$ such that $\overline{\Gamma} \vdash \phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}) \sim \phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}) : \mathtt{LL} \to c'$. Thus there exists $c'' \subseteq c'$ such that $\overline{\Gamma} \vdash \mathtt{sign}(M, k) \sim \mathtt{sign}(N, k') : \mathtt{LL} \to c''$. Hence by Lemma 18, there exists $c''' \subseteq c'' \subseteq c'$ such that $\overline{\Gamma} \vdash M \sim N : \mathtt{LL} \to c'''$. Moreover M, N are ground, since by assumption $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})$ and $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$ restricted to $\mathrm{vars}(R)$ are ground. Therefore, by Lemma 26, there exists a recipe R' without destructors such that $M = R'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell(c'''))$ and $N = R'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r(c'''))$. Since $c''' \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}$, this proves the claim for this case.
- In the second case, there exist S'', K' such that $S' = \mathtt{sign}(S'', K')$. Since $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \mathtt{sign}(M, k)$, we have $S''(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = M$. Hence $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = M = S''(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, and similarly for $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$. Moreover, S'' being a subterm of S' it also satisfies the conditions on the domains, and thus the property holds with $R' = S''$.

– If $R = \pi_1(S)$ for some recipe S then since $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$, we have $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \langle M_1, M_2 \rangle$, where $M_1 = R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$, and M_2 is a message.

Similarly, $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \langle N_1, N_2 \rangle$, where $N_1 = R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$, and N_2 is a message.

Since $S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \langle M_1, M_2 \rangle \neq \bot$, by the induction hypothesis, there exists S' such that $\mathrm{vars}(S') \subseteq \mathrm{vars}(S)$, $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \langle M, k \rangle$, and $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) = S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \langle N, k \rangle$.

Since $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \langle M_1, M_2 \rangle$, it is clear from the definition of $\downarrow$ that either $S' = x$ for some $x \in \mathcal{AX}$, or $S' = \langle S_1, S_2 \rangle$ for some S_1, S_2.

The first case is impossible, since by Lemma 35, $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$, and thus $[\![\overline{c}]\!]_{\sigma,\sigma'}$ does not contain pairs.

In the second case, there exist S_1, S_2 such that $S' = \langle S_1, S_2 \rangle$. Since $S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \langle M_1, M_2 \rangle$, we have $S_1(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = M_1$. Hence $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = M_1 = S_1(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, and similarly for $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$. Moreover, S_1 being a subterm of S' it also satisfies the conditions on the domains, and thus the property holds with $R' = S_1$.

– If $R = \pi_2(S)$ for some S: this case is similar to the π_1 case.

Lemma 39. *For all term t and substitution σ containing only messages, if $t\downarrow \neq \bot$, then $(t\sigma)\downarrow = (t\downarrow)\sigma$.*

Proof. This property is easily proved by induction on t. In the base case where t is a variable x, by definition of $\downarrow$, since $\sigma(x)$ is a messages, $\sigma(x)\downarrow = \sigma(x)$ and the claim holds. In the other base cases where t is a name, key or constant the claim trivially holds. We prove the case where t starts with a constructor other than `enc`, `aenc`, `sign` generically for $t = f(t_1, t_2)$. We then have $t_1\sigma\downarrow \neq \bot$ and $t_2\sigma\downarrow \neq \bot$, and $t\sigma\downarrow = f(t_1\sigma\downarrow, t_2\sigma\downarrow)$, which, by the induction hypothesis, is equal to $f(t_1\downarrow\sigma, t_2\downarrow\sigma)$, *i.e.* to $f(t_1, t_2)\downarrow\sigma$. The case where f is `enc`, `aenc` or `sign` is similar, but we in addition know that $t_2\downarrow$ is a key.

Finally if t starts with a destructor, $t = d(t_1, t_2)$, we know that $t_1\downarrow$ starts with the corresponding constructor f: $t_1\downarrow = f(t_3, t_4)$. In the case of encryptions and signatures we know in addition that $t_4\downarrow$ and $t_2\downarrow$ are the same key (resp. public key/verification key). We then have $t\downarrow = t_3\downarrow$, and $t\sigma\downarrow = t_3\sigma\downarrow$ (or t_4 in the case of the second projection π_2). Hence by the induction hypothesis, $t\sigma\downarrow = t_3\downarrow\sigma = t\downarrow\sigma$ and the claim holds.

Lemma 40. *For all ground σ, σ', for all recipe R such that*

- $\exists \Gamma' \subseteq \overline{\Gamma}.\ \exists c_\sigma.\ \Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c_\sigma$,
- $\mathrm{vars}(R) \subseteq \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell(\overline{c}))$,
- $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})$ *and* $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$ *restricted to* $\mathrm{vars}(R)$ *are ground,*

for all $x \in \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell(\overline{c}))$, if $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = (\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x)$ then R is a variable $y \in \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell(\overline{c}))$, or $R \in \mathcal{C} \cup \mathcal{FN}$.

Similarly, if $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}) = (\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))(x)$ then R is a variable $y \in \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r(\overline{c}))$ or $R \in \mathcal{C} \cup \mathcal{FN}$.

Proof. We only detail the proof for $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})$, as the proof for $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$ is similar.

We distinguish several cases for R.

- If $R = a \in \mathcal{C} \cup \mathcal{FN}$: the claim clearly holds.
- If $R = x \in \mathcal{AX}$ then the claim trivially holds.
- If $R = \mathtt{enc}(S, K)$ or $\mathtt{sign}(S, K)$ for some recipes S, K: these two cases are similar, we only detail the encryption case. $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x)$ is then an encrypted message. By Lemma 35, $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$. Hence there exist $k, k' \in \mathcal{K}$ and T such that $K(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = k$ and $\Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$. This is only possible if there exists a variable z such that $K = z$ and $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\tilde{c}]\!]_{\sigma,\sigma'}))(z) = k$. Since $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$, z can only be in $\mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}})$. By definition of $\phi^{\overline{\Gamma}}_{\mathrm{LL}}$, this implies that $\Gamma(k, k) <: \mathrm{key}^{\mathrm{LL}}(T')$ for some T'. Since $\overline{\Gamma}$ is well-formed, this contradicts $\Gamma(k, k') <: \mathrm{key}^{\mathtt{HH}}(T)$: this case is impossible.
- If $R = \mathtt{aenc}(S, K)$ or $\mathtt{h}(S)$ for some recipes S, K: these two cases are similar, we only detail the encryption case. $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x)$ is then an asymmetrically encrypted message. By Lemma 35, we know that $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$. Hence $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$ contains directly under pairs a nonce n such that $\overline{\Gamma}(n) = \tau_n^{\mathtt{HH},a}$, or a key $k \in \mathcal{K}$ such that $\overline{\Gamma}(k, k') = \mathrm{key}^{\mathtt{HH}}(T)$ for some T, k'.
  This is only possible if there exists a recipe S' such that $S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = n$ (resp. k). Since S' can only contain names from $\mathcal{FN}$ (and no keys), this implies that there exists a variable z such that $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\tilde{c}]\!]_{\sigma,\sigma'}))(z) = n$ (resp. k). As $\mathtt{step2}_{\overline{\Gamma}}(\overline{c}) = \mathtt{true}$, z can only be in $\mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}})$. Thus, by definition of $\phi^{\overline{\Gamma}}_{\mathrm{LL}}$, $\overline{\Gamma}(n) = \tau_n^{\mathrm{LL},a}$ for some a (resp. $\overline{\Gamma}(k, k) <: \mathrm{key}^{\mathrm{LL}}(T')$ for some T'), which is contradictory (as $\overline{\Gamma}$ is well-formed).
- Finally, the head symbol of R cannot be $\langle\cdot,\cdot\rangle$, `dec`, `adec`, `checksign`, π_1, π_2 since $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$ by Lemma 35.

Lemma 41. *For all ground σ, σ', for all recipes R, S such that*

- $\exists \Gamma' \subseteq \overline{\Gamma}.\ \exists c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}.\ \Gamma'_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : \Gamma'_{\mathcal{X}} \to c_\sigma$,
- $\mathrm{vars}(R) \cup \mathrm{vars}(S) \subseteq \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell(\overline{c}))$,
- $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})$ *and* $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})$ *restricted to* $\mathrm{vars}(R) \cup \mathrm{vars}(S)$ *are ground,*

we have

$$(R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow = (S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow \iff (R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow = (S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow .$$

Proof. We only detail the proof for the $(\Rightarrow)$ direction, as the other direction is similar. We then assume that

$$(R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow = (S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow .$$

Let us first note that

$$(R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow \neq \bot \quad \iff \quad (R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))) \downarrow \neq \bot.$$

This follows from Lemmas 37 and 23.

Similarly, we have

$$S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot \quad \iff \quad S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot.$$

Therefore, if $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \bot$, then $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \bot$. By assumption, in that case we also have $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \bot$, and thus $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = \bot$, and the claim holds.

Let us now assume that $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$, *i.e.*, by assumption, that $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$. We then have $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$ and $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow \neq \bot$.

By Lemma 38, there exist recipes R', S' without destructors such that

- $\mathrm{vars}(R') \cup \mathrm{vars}(S') \subseteq \mathrm{dom}(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$,
- $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$,
- $R(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$.
- $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$,
- $S(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$.

By the assumption, we have $R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$.

We show that $R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$ by induction on the recipes R', S'. Since $R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, we can distinguish four cases for R' and S'.

- If they have the same head symbol, either this symbol is a nonce or constant and the claim is trivial, or it is a variable, and we handle this case later, or it is a constructor. We write the proof for this last case generically for $R' = f(R'')$ and $S' = f(S'')$. We have necessarily $R''(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S''(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$. It then follows by applying the induction hypothesis to R'' and S'' that $R''(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) = S''(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))$. The claim follows by applying f on both sides of this equality.
- If R' is a variable and not S': then by Lemma 40, $S' \in \mathcal{C} \cup \mathcal{FN}$. Let us denote $R' = x$. By Lemma 37, there exists c_x such that $\overline{\Gamma} \vdash (\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) \sim (\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) : \mathtt{LL} \to c_x$. Since $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) = R'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) \in \mathcal{C} \cup \mathcal{FN}$, by Lemma 20, we have $(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) = S'(\phi^{\overline{\Gamma}}_{\mathrm{LL}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, *i.e.* $R'(\phi^{\Gamma}_{\mathrm{LL}} \cup \phi_r(\tilde{c})) = S'(\phi^{\Gamma}_{\mathrm{LL}} \cup \phi_r(\tilde{c}))$.

– If S' is a variable and not R': this case is similar to the previous one.
– If R', S' are two variables x and y, we have $(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) = (\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))(y)$. We can then prove $(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))(x) = (\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'}))(y)$. Indeed:
  - if $x, y \in \mathrm{dom}(\phi_{\mathrm{LL}}^{\overline{\Gamma}})$, this follows from the definition of $\phi_{\mathrm{LL}}^{\overline{\Gamma}}$.
  - if $x \in \mathrm{dom}(\phi_{\mathrm{LL}}^{\overline{\Gamma}})$ and $y \in \mathrm{dom}(\phi_\ell([\![\tilde{c}]\!]_{\sigma,\sigma'}))$: then by definition of $\phi_{\mathrm{LL}}^{\overline{\Gamma}}$, $R'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = \phi_{\mathrm{LL}}^{\overline{\Gamma}}(x)$ is a nonce, key, public key, or verification key. Hence $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})(y)$ is also a nonce, key, public key or verification key. This is not possible, as by Lemma 35, $\mathtt{step2}_{\overline{\Gamma}}([\![\overline{c}]\!]_{\sigma,\sigma'}) = \mathtt{true}$.
  - if $x, y \in \mathrm{dom}(\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$: then there exist $M \sim M' \in \overline{c}$, $N \sim N' \in \overline{c}$ such that $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})(x) = M\sigma$, $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})(x) = M'\sigma'$, $\phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})(y) = N\sigma$, $\phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})(y) = N'\sigma'$. Since $M\sigma = N\sigma$, M, N are unifiable, let μ be their most general unifier. There exists θ such that $\sigma = \mu\theta$.
    Let then α be the restriction of μ to $\{x \in \mathrm{vars}(M) \cup \mathrm{vars}(N) \mid \overline{\Gamma}(x) = \mathtt{LL} \ \wedge \ \mu(x) \in \mathcal{N} \text{ is a nonce}\}$.
    By $\mathtt{step3}_{\overline{\Gamma}}(\overline{c})$, we have $M'\alpha = N'\alpha$.

    Note that α and β have disjoint domains. Let $x \in \mathrm{dom}(\alpha) \cap (\mathrm{vars}(M') \cup \mathrm{vars}(N'))$. Then $\mu(x) = n$ for some $n \in \mathcal{N}$. Thus $\sigma(x) = \mu(x)\theta = n$.
    Since, by assumption, $\exists \Gamma' \subseteq \overline{\Gamma}.\ \exists c_\sigma.\ {\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c_\sigma$, there exists c_x such that ${\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash n \sim \sigma'(x) : \mathtt{LL} \to c_x$. Hence by Lemma 20, $\sigma'(x) = n = \mu(x) = \alpha(x) = (\alpha\sigma')(x)$.
    We have shown that for all $x \in \mathrm{dom}(\alpha) \cap (\mathrm{vars}(M') \cup \mathrm{vars}(N'))$, $(\alpha\sigma')(x) = \sigma'(x)$. Thus, on $\mathrm{dom}(\alpha) \cap (\mathrm{vars}(M') \cup \mathrm{vars}(N'))$, we have $\alpha\sigma' = \sigma'$.

    Therefore, since $M'\alpha = N'\alpha$, we have $M'\alpha\sigma' = N'\alpha\sigma'$, *i.e.* $M'\sigma' = N'\sigma'$. This proves the claim.

Finally, since $R'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'})) = S'(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![\overline{c}]\!]_{\sigma,\sigma'}))$, we have $R(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow = S(\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![\overline{c}]\!]_{\sigma,\sigma'})) \downarrow$.
This proves the property.

Lemma 42. *For all ground σ, σ', for all $\Gamma' \subseteq \overline{\Gamma}$, and for all $c' \subseteq \overline{c}$, such that*

– $\exists c_\sigma \subseteq [\![\overline{c}]\!]_{\sigma,\sigma'}.\ {\Gamma'}_{\mathcal{N},\mathcal{K}} \vdash \sigma \sim \sigma' : {\Gamma'}_{\mathcal{X}} \to c_\sigma$,
– ${\Gamma'}_{\mathcal{N},\mathcal{K}} = \overline{\Gamma}_{\mathcal{N},\mathcal{K}}$ *and* $\mathrm{vars}(c') \subseteq \mathrm{dom}(\Gamma')$,

the frames $\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_\ell([\![c']\!]_{\sigma,\sigma'})$ and $\phi_{\mathrm{LL}}^{\overline{\Gamma}} \cup \phi_r([\![c']\!]_{\sigma,\sigma'})$ are statically equivalent.

Proof. This is a direct consequence of Lemma 41, by unfolding the definition of static equivalence.

Lemma 43. *c is consistent in Γ.*

Proof. By Lemma 34, it suffices to show that $\overline{c}$ is consistent in $\overline{\Gamma}$. This is a direct consequence of Lemma 42, by unfolding the definition of consistency.

This next theorem corresponds to Theorem 2.

Theorem 9 (Soundness of the procedure). *Let C be a constraint set without infinite nonce types,* i.e.

$$\forall (c, \Gamma) \in C.\ \forall l, l', m, n. \Gamma(x) \neq [\![\tau_m^{l,\infty}\ ;\ \tau_n^{l',\infty}]\!].$$

If $\mathtt{check\_const}(C)$ *succeeds, then C is consistent.*

Proof. The previous lemmas directly imply that for all $(c, \Gamma) \in C$, c is consistent in Γ. This proves the theorem.

B.4 Consistency for replicated processes

In this subsection, we prove the results regarding the procedure when checking consistency in the replicated case.

In this subsection, we only consider constraints obtained by typing processes (with the same key types). Notably, by the well-formedness assumptions on the processes, this means that a nonce n is always associated with the same nonce type.

Theorem 10. *Let C and C' be two constraint sets that*

- *do not share any common variable,* i.e.

$$\forall (c, \Gamma) \in C.\ \forall (c', \Gamma') \in C'.\ \mathrm{dom}(\Gamma_{\mathcal{X}}) \cap \mathrm{dom}(\Gamma'_{\mathcal{X}}) = \emptyset;$$

- *only share nonces which have the same finite nonce type,* i.e.

$$\forall (c, \Gamma) \in C.\ \forall (c', \Gamma') \in C'.\ \forall m \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma') \cap \mathcal{N}.\ \exists l.\ \Gamma(m) = \Gamma'(m) = \tau_m^{l,1};$$

- *only share keys which are paired in the same way and have the same types,* i.e.

$$\begin{aligned}&\forall (c, \Gamma) \in C.\ \forall (c', \Gamma') \in C'.\ \forall k \in \mathrm{keys}(\Gamma) \cap \mathrm{keys}(\Gamma').\forall k' \in \mathcal{K}.\\ &\quad ((k, k') \in \mathrm{dom}(\Gamma) \Leftrightarrow (k, k') \in \mathrm{dom}(\Gamma'))\ \wedge\ ((k', k) \in \mathrm{dom}(\Gamma) \Leftrightarrow (k', k) \in \mathrm{dom}(\Gamma'))\end{aligned}$$

  and

$$\forall (c, \Gamma) \in C.\ \forall (c', \Gamma') \in C'.\ \forall (k, k') \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Gamma').\ \Gamma(k, k') = \Gamma'(k, k').$$

  For all $n \in \mathbb{N}$, if $\texttt{check\_const}([\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n) = \texttt{true}$*, then*

$$\texttt{check\_const}(\cup_{\times\, 1 \leq i \leq n} [\,C\,]_i^n) \cup_\times [\,C'\,]_1^n) = \texttt{true}.$$

Proof. Let $n \in \mathbb{N}$. Let C, C' be as defined in the statement of the theorem.

Let $(c, \Gamma) \in (\cup_{\times\, 1 \leq i \leq n} [\,C\,]_i^n) \cup_\times [\,C'\,]_1^n$. By definition of $\cup_\times$, there exists $(c', \Gamma') \in [\,C'\,]_1^n$, and for all $i \in [\![1, n]\!]$, there exists $(c_i, \Gamma_i) \in [\,C\,]_i^n$, such that

- $c = (\cup_{1 \leq i \leq n} c_i) \cup c'$;
- $\Gamma = (\bigcup_{1 \leq i \leq n} \Gamma_i) \cup \Gamma'$.

Let $i \in [\![1, n]\!]$. Since $(c_i, \Gamma_i) \in [\,C\,]_i^n$, by definition of $[\,C\,]_i^n$ there exists $(c'_i, \Gamma'_i) \in C$ such that

- $c_i = [\,c'_i\,]_i^{\Gamma_i}$;
- $\Gamma_i \in \mathrm{branches}([\,\Gamma'_i\,]_i^n)$.

Note that this implies $\mathrm{dom}(\Gamma_{i\mathcal{X}})$ only contains variables indexed by i, and, from the assumption that $\mathrm{vars}(c'_i) \subseteq \mathrm{dom}(\Gamma'_{i\mathcal{X}})$, that $\mathrm{vars}(c_i) \subseteq \mathrm{dom}(\Gamma_{i\mathcal{X}})$.

For all $i \in [\![1, n]\!]$, let δ_i^1 denote the function on terms which consists in exchanging all occurrences of the indices i and 1, *i.e.* replacing any occurrence of m_i (for all nonce m with an infinite nonce type) with m_1, any occurrence of m_1 with m_i, any occurrence of k_1 with k_i (for all key k), any occurrence of k_i with k_1, any occurrence of x_i with x_1 (for all variable x), and any occurrence of x_1 with x_i (also for all variable x).

We extend this function to constraints, types, typing environments and constraint sets. In the case of types we use it to denote the replacement of nonces appearing in the refinements. In the case of typing environments it denotes the replacement of nonces appearing in the types, and of nonces, keys and variables in the domain of

the environment, *i.e.* $(\delta_i^1(\Gamma))(x_1) = \delta_i^1(\Gamma(x_i))$. In the case of constraint sets it denotes the application of the function to each constraint and environment in the constraint set.

Similarly, we denote δ_i^2 the function exchanging indices i and 2.

For all $h \in [\![1, n]\!]$ and all $i \neq j \in [\![1, n]\!]$, such that $i \neq 2$ and $j \neq 1$, let

$$(c_h^{i,j}, \Gamma_h^{i,j}) = (c_h, \Gamma_h)\delta_i^1\delta_j^2.$$

Similarly, for all $h \in [\![1, n]\!]$ and all $i \in [\![1, n]\!]$, let

$$(c_h^{i,i}, \Gamma_h^{i,i}) = (c_h, \Gamma_h)\delta_i^1.$$

Finally, for all $i \in [\![1, n]\!]$, let Γ'^i be the typing environment such that $\mathrm{dom}(\Gamma'^i) = \mathrm{dom}(\Gamma')$ and $\forall x \in \mathrm{dom}(\Gamma'^i)$. $\Gamma'^i(x) = \Gamma'(x)\delta_i^1$.

Since $(c_i, \Gamma_i) \in [\, C \,]_i^n$, we can show that $(c_i^{i,j}, \Gamma_i^{i,j}) \in [\, C \,]_1^n$. Indeed, recall that there exists $(c_i', \Gamma_i') \in C$ such that $c_i = [\, c_i' \,]_i^{\Gamma_i}$ and $\Gamma_i \in \mathrm{branches}([\, \Gamma_i' \,]_i^n)$. c_i only contains variables, keys and names indexed by i or unindexed, hence it is clear that $c_i^{i,j} = [\, c_i' \,]_1^{\Gamma_i}$. Moreover, since $\Gamma_i \in \mathrm{branches}([\, \Gamma_i' \,]_i^n)$, it is clear that $\Gamma_i^{i,j} \in \mathrm{branches}([\, \Gamma_i' \,]_i^n \delta_i^1 \delta_j^2)$. By definition, indexed nonces, variables, or keys appear in $[\, \Gamma_i' \,]_i^n$ only in its domain, and as parts of union types of the form $[\![\tau_{m_1}^{l,1} ; \tau_{p_1}^{l',1}]\!] \vee \ldots \vee [\![\tau_{m_n}^{l,1} ; \tau_{p_n}^{l',1}]\!]$. This union type is left unchanged by $\delta_i^1\delta_j^2$: since $i \neq j$, $i \neq 2$, and $j \neq 1$, $\delta_i^1\delta_j^2$ is indeed only performing a permutation of the indices. Hence, $[\, \Gamma_i' \,]_i^n \delta_i^1 \delta_j^2 = [\, \Gamma_i' \,]_1^n$. Thus $\Gamma_i^{i,j} \in \mathrm{branches}([\, \Gamma_i' \,]_1^n)$. Therefore, $(c_i^{i,j}, \Gamma_i^{i,j}) \in [\, C \,]_1^n$.

Note that $\mathrm{dom}(\Gamma_i^{i,j})$ only contains variables indexed by 1; and that, since $\mathrm{vars}(c_i) \subseteq \mathrm{dom}(\Gamma_i)$, we have $\mathrm{vars}(c_i^{i,j}) \subseteq \mathrm{dom}(\Gamma_i^{i,j})$.

Similarly, if $j \neq i$, $i \neq 2$ and $j \neq 1$, $(c_j^{i,j}, \Gamma_j^{i,j}) \in [\, C \,]_2^n$. Note that $\mathrm{dom}(\Gamma_j^{i,j})$ only contains variables indexed by 2; and that $\mathrm{vars}(c_j^{i,j}) \subseteq \mathrm{dom}(\Gamma_j^{i,j})$.

Similarly, we also have $(c', \Gamma'^i) \in [\, C' \,]_1^n$.

By assumption, for all $(c, \Gamma) \in C$ and for all $(c', \Gamma') \in C'$, $\mathrm{dom}(\Gamma_\mathcal{X}) \cap \mathrm{dom}(\Gamma'_\mathcal{X}) = \emptyset$, and Γ, Γ' give the same types to the nonces and keys they have in common. Hence for all $(c'', \Gamma'') \in [\, C \,]_1^n$, and all $(c''', \Gamma''') \in [\, C' \,]_1^n$, we know that Γ'' and Γ''' are compatible. In particular this applies to all the $\Gamma_i^{i,j}$ and Γ' (as well as $\Gamma_i^{i,j}$ and Γ'^i).

Moreover, for all $(c'', \Gamma'') \in [\, C \,]_2^n$, and all $(c''', \Gamma''') \in [\, C' \,]_1^n$, since $\mathrm{dom}(\Gamma''') \subseteq \{x_1 \mid x \in \mathcal{X}\}$, and $\mathrm{dom}(\Gamma'') \subseteq \{x_2 \mid x \in \mathcal{X}\}$, Γ'' and Γ''' are also compatible. This in particular applies to $\Gamma_j^{i,j}$ for all $i \neq j \in [\![1, n]\!]$ and Γ' (as well as $\Gamma_j^{i,j}$ and Γ'^i).

If C is empty, then so is $\cup_{\times 1 \leq i \leq n} [\, C \,]_i^n$ and the claim clearly holds. Let us now assume that C is not empty. Hence for all $i \in [\![1, n]\!]$, $[\, C \,]_i^n$ is not empty.

The procedure for c, Γ is as follows:

1. We compute $\mathtt{step1}_\Gamma(c)$. Following the notations used in the procedure, we have

$$F = \{x \in \mathrm{dom}(\Gamma) \mid \exists m, n, l, l'.\ \Gamma(x) = [\![\tau_m^{l,1} ; \tau_n^{l',1}]\!]\},$$

and we write $(\overline{c}, \overline{\Gamma}) \stackrel{\mathrm{def}}{=} \mathtt{step1}_\Gamma(c)$.

For all $i \in [\![1, n]\!]$, let $(\overline{c_i}, \overline{\Gamma}_i) \stackrel{\mathrm{def}}{=} \mathtt{step1}_{\Gamma_i}(c_i)$. Let also $(\overline{c}', \overline{\Gamma}') \stackrel{\mathrm{def}}{=} \mathtt{step1}_{\Gamma'}(c')$. We have $\overline{c} = (\cup_{1 \leq i \leq n} \overline{c_i}) \cup \overline{c}'$, and $\overline{\Gamma} = (\cup_{1 \leq i \leq n} \overline{\Gamma}_i) \cup \overline{\Gamma}'$.

For all $h, i, j \in [\![1, n]\!]$, such that either $i \neq j$ and $i \neq 2$ and $j \neq 1$, or $i = j$, let us also denote $(\overline{c}_h^{i,j}, \overline{\Gamma}_h^{i,j}) \stackrel{\text{def}}{=} \texttt{step1}_{\Gamma_h^{i,j}}(c_h^{i,j})$. Similarly, for all $i \in [\![1, n]\!]$, let also $(\overline{c'}^i, \overline{\Gamma'}^i) \stackrel{\text{def}}{=} \texttt{step1}_{\Gamma'^i}(c')$. Since, for $i \neq j$, $(c_h^{i,j}, \Gamma_h^{i,j}) = (c_h, \Gamma_h)\delta_i^1\delta_j^2$, it can easily be shown (by induction on the terms) that $(\overline{c}_h^{i,j}, \overline{\Gamma}_h^{i,j}) = (\overline{c_h}, \overline{\Gamma}_h)\delta_i^1\delta_j^2$. Similarly, $(\overline{c}_h^{i,i}, \overline{\Gamma}_h^{i,i}) = (\overline{c_h}, \overline{\Gamma}_h)\delta_i^1$. Finally, we similarly also have $(\overline{c'}^i, \overline{\Gamma'}^i) = (\overline{c}', \overline{\Gamma}')\delta_p^1$.

2. We check that $\texttt{step2}_{\overline{\Gamma}}(\overline{c})$ holds, *i.e.* that each $M \sim N \in \overline{c}$ has the correct form (with respect to the definition of `step2`).
   If $M \sim N \in \overline{c}$, either $M \sim N \in \overline{c}'$, or there exists $i \in [\![1, n]\!]$ such that $M \sim N \in \overline{c_i}$.
   - In the first case, $M \sim N \in \overline{c}'$. By assumption, $[\, C \,]_1^n$ and $[\, C \,]_2^n$ are not empty. Hence there exist $(c'', \Gamma'') \in [\, C \,]_1^n$ and $(c''', \Gamma''') \in [\, C \,]_2^n$. Thus, $(c'' \cup c''' \cup c', \Gamma'' \cup \Gamma''' \cup \Gamma') \in [\, C \,]_1^n \cup_\times [\, C \,]_2^n \cup_\times [\, C' \,]_1^n$ (as noted previously, Γ'', Γ''', and Γ' are compatible). Hence, by assumption, $\texttt{check\_const}(\{(c'' \cup c''' \cup c', \Gamma'' \cup \Gamma''' \cup \Gamma')\})$ succeeds.
     If $\overline{c}'' = fst(\texttt{step1}_{\Gamma''}(c''))$, and $\overline{c}''' = fst(\texttt{step1}_{\Gamma'''}(c'''))$, then we have

     $$\overline{c}'' \cup \overline{c}''' \cup \overline{c}' = fst(\texttt{step1}_{\Gamma'' \cup \Gamma''' \cup \Gamma'}(c'' \cup c''' \cup c')).$$

     Therefore, $\texttt{step2}_{\overline{\Gamma}}(\tilde{c}'' \cup \tilde{c}''' \cup \tilde{c}') = \texttt{true}$.
     In particular, $M \sim N \in \overline{c}'$ has the correct form.
   - In the second case, $M \sim N \in \overline{c_i}$ for some $i \in [\![1, n]\!]$.
     Let $M' = M\delta_i^1$ and $N' = N\delta_i^1$. Since $\overline{c}_i^{i,i} = \overline{c_i}\delta_i^1$, we have $M' \sim N' \in \overline{c}_i^{i,i}$.
     By assumption, $[\, C \,]_2^n$ is not empty, hence there exists $(c'', \Gamma'') \in [\, C \,]_2^n$. Thus, $(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma') \in [\, C \,]_1^n \cup_\times [\, C \,]_2^n \cup_\times [\, C' \,]_1^n$ (as noted previously, $\Gamma_i^{i,i}$, Γ'', and Γ' are compatible). Hence, by assumption, $\texttt{check\_const}(\{(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma')\})$ succeeds. If $\overline{c}'' = fst(\texttt{step1}_{\Gamma''}(c''))$, then $\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c}' = fst(\texttt{step1}_{\Gamma_{i,1}'' \cup \Gamma'' \cup \Gamma'}(c_i^{i,i} \cup c'' \cup c'))$. Therefore, $\texttt{step2}_{\overline{\Gamma}}(\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c}')$ holds.
     In particular, $M' \sim N' \in \tilde{c}_i^{i,i}$, has the correct form. By examining all the cases and using the fact that for all m_i, m_j, if m_i is associated with the type $\tau_{m_i}^{l,a}$ and m_j with $\tau_{m_j}^{l',a}$ then $l = l'$, and similarly for keys; it follows that $M \sim N$ also has the correct form.

   Therefore, $\texttt{step2}_{\overline{\Gamma}}(\overline{c})$ holds.

3. Finally, we check that $\texttt{step3}_{\overline{\Gamma}}(\overline{c})$ holds. Let $M_1 \sim N_1 \in \overline{c}$ and $M_2 \sim N_2 \in \overline{c}$. Let us prove the property in the case where M_1 and M_2 are unifiable with a most general unifier μ. The case where N_1 and N_2 are unifiable is similar.
   Let then α be the restriction of μ to $\{x \in \text{vars}(M_1) \cup \text{vars}(M_2) \mid \overline{\Gamma}(x) = \texttt{LL} \ \wedge \ \mu(x) \in \mathcal{N}\}$.
   We have to prove that $N_1\alpha = N_2\alpha$.

   Since we already have $\overline{c} = (\cup_{1 \leq i \leq n} \overline{c_i}) \cup \overline{c}'$, we know that:
   - either there exists $i \in [\![1, n]\!]$ such that $M_1 \sim N_1 \in \overline{c_i}$;
   - or $M_1 \sim N_1 \in \overline{c}'$;

   and
   - either there exists $j \in [\![1, n]\!]$ such that $M_2 \sim N_2 \in \overline{c_j}$;
   - or $M_2 \sim N_2 \in \overline{c}'$.

   Let us first prove the case where there exist $i, j \in [\![1, n]\!]$ such that $M_1 \sim N_1 \in \overline{c_i}$ and $M_2 \sim N_2 \in \overline{c_j}$. We distinguish two cases.

– if $i \neq j$: The property to prove is symmetric between $M_1 \sim N_1 \in \overline{c}$ and $M_2 \sim N_2 \in \overline{c}$. Hence without loss of generality we may assume that $i \neq 2$ and $j \neq 1$.
Let then $M'_1 = M_1\delta_i^1\delta_j^2$, $N'_1 = N_1\delta_i^1\delta_j^2$, $M'_2 = M_2\delta_i^1\delta_j^2$, $N'_2 = N_2\delta_i^1\delta_j^2$.
Since $\overline{c}_i^{i,j} = \overline{c_i}\delta_i^1\delta_j^2$, we have $M'_1 \sim N'_1 \in \overline{c}_i^{i,j}$. Similarly, $M'_2 \sim N'_2 \in \overline{c}_j^{i,j}$.
Since M_1 and M_2 are unifiable, then so are M'_1 and M'_2, with a most general unifier μ' which satisfies $\mu(x) = t \Leftrightarrow \mu'(x\delta_i^1\delta_j^2) = t\delta_i^1\delta_j^2$.
Let then α' be the restriction of μ' to $\{x \in \mathrm{vars}(M'_1) \cup \mathrm{vars}(M'_2) \mid (\overline{\Gamma}_i^{i,j} \cup \overline{\Gamma}_j^{i,j})(x) = \mathtt{LL} \wedge \mu'(x) \in \mathcal{N} \text{ is a nonce}\}$.
Similarly α' is such that $\forall x \in \mathrm{dom}(\alpha').\forall n.\ \alpha(x) = n \Leftrightarrow \alpha'(x\delta_i^1\delta_j^2) = n\delta_i^1\delta_j^2$, *i.e.* $\delta_i^1\delta_j^2\alpha'\delta_i^1\delta_j^2 = \alpha$.
By assumption, $\mathtt{check\_const}(\{(c_i^{i,j} \cup c_j^{i,j} \cup c', \Gamma_i^{i,j} \cup \Gamma_j^{i,j} \cup \Gamma')\})$ succeeds since $(c_i^{i,j} \cup c_j^{i,j} \cup c', \Gamma_i^{i,j} \cup \Gamma_j^{i,j} \cup \Gamma') \in [\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n$.
Thus, $\mathtt{step3}_{\overline{\Gamma}_i^{i,j} \cup \overline{\Gamma}_j^{i,j} \cup \overline{\Gamma}'}(\overline{c}_i^{i,j} \cup \overline{c}_j^{i,j} \cup \overline{c}')$ holds, and since $\{M'_1 \sim N'_1, M'_2 \sim N'_2\} \subseteq \overline{c}_i^{i,j} \cup \overline{c}_j^{i,j} \cup \overline{c}'$, we know that since M'_1, M'_2 are unifiable, $N'_1\alpha' = N'_2\alpha'$.
Thus $N'_1\alpha'\delta_i^1\delta_j^2 = N'_2\alpha'\delta_i^1\delta_j^2$, *i.e.*, since $i \neq j$, and $\delta_i^1\delta_j^2\alpha'\delta_i^1\delta_j^2 = \alpha$, $N_1\alpha = N_2\alpha$. Therefore the claim holds in this case.
– if $i = j$ then let $M'_1 = M_1\delta_i^1$, $N'_1 = N_1\delta_i^1$, $M'_2 = M_2\delta_i^1$, $N'_2 = N_2\delta_i^1$. Since $\overline{c}_i^{i,i} = \overline{c_i}\delta_i^1$, we have $M'_1 \sim N'_1 \in \overline{c}_i^{i,i}$. Similarly, $M'_2 \sim N'_2 \in \overline{c}_i^{i,i}$.
Since M_1 and M_2 are unifiable, then so are M'_1 and M'_2, with a most general unifier μ' which satisfies $\mu(x) = t \Leftrightarrow \mu'(x\delta_i^1) = t\delta_i^1$.
Let then α' be the restriction of μ' to $\{x \in \mathrm{vars}(M'_1) \cup \mathrm{vars}(M'_2) \mid \overline{\Gamma}_i^{i,i}(x) = \mathtt{LL} \wedge \mu'(x) \in \mathcal{N} \text{ is a nonce}\}$.
Similarly α' is such that $\forall x \in \mathrm{dom}(\alpha').\forall n.\ \alpha(x) = n \Leftrightarrow \alpha'(x\delta_i^1) = n\delta_i^1$, *i.e.* $\delta_i^1\alpha'\delta_i^1 = \alpha$.
By assumption, $[\,C\,]_2^n$ is not empty, hence there exists $(c'', \Gamma'') \in [\,C\,]_2^n$. Thus, $(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma') \in [\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n$. Hence, by assumption, $\mathtt{check\_const}(\{(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma')\})$ succeeds. If $\overline{c}'' = fst(\mathtt{step1}_{\Gamma''}(c''))$, then $\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c}' = fst(\mathtt{step1}_{\Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma'}(c_i^{i,i} \cup c'' \cup c'))$.
Thus, $\mathtt{step3}_{\overline{\Gamma}_i^{i,i} \cup \overline{\Gamma}'' \cup \overline{\Gamma}'}(\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c}')$ holds, and since $\{M'_1 \sim N'_1, M'_2 \sim N'_2\} \subseteq \overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c}'$, we know that $N'_1\alpha' = N'_2\alpha'$.
Thus $N'_1\alpha'\delta_i^1 = N'_2\alpha'\delta_i^1$, *i.e.*, since $\delta_i^1\alpha'\delta_i^1$, $N_1\alpha = N_2\alpha$. Therefore the claim holds in this case.

Let us now prove the case where there exists $i \in [\![1, n]\!]$ such that $M_1 \sim N_1 \in \overline{c_i}$, and $M_2 \sim N_2 \in \overline{c}'$. The symmetric case, where $M_1 \sim N_1 \in \overline{c}'$ and there exists $j \in [\![1, n]\!]$ such that $M_2 \sim N_2 \in \overline{c_j}$, is similar.
Let then $M'_1 = M_1\delta_i^1$, $N'_1 = N_1\delta_i^1$, $M'_2 = M_2\delta_i^1$, $N'_2 = N_2\delta_i^1$. Since $\overline{c}_i^{i,i} = \overline{c_i}\delta_i^1$, we have $M'_1 \sim N'_1 \in \overline{c}_i^{i,i}$. Similarly, $M'_2 \sim N'_2 \in \overline{c'}^i$.
Since M_1 and M_2 are unifiable, then so are M'_1 and M'_2, with a most general unifier μ' which satisfies $\mu(x) = t \Leftrightarrow \mu'(x\delta_i^1) = t\delta_i^1$.
Let then α' be the restriction of μ' to $\{x \in \mathrm{vars}(M'_1) \cup \mathrm{vars}(M'_2) \mid (\overline{\Gamma}_i^{i,i} \cup \Gamma'^i)(x) = \mathtt{LL} \wedge \mu'(x) \in \mathcal{N} \text{ is a nonce}\}$.
Similarly α' is such that $\forall x \in \mathrm{dom}(\alpha').\forall n.\ \alpha(x) = n \Leftrightarrow \alpha'(x\delta_i^1) = n\delta_i^1$, *i.e.* $\delta_i^1\alpha'\delta_i^1 = \alpha$.
By assumption, $[\,C\,]_2^n$ is not empty, hence there exists $(c'', \Gamma'') \in [\,C\,]_2^n$. Moreover, as noted previously, $(c', \Gamma'^i) \in [\,C'\,]_1^n$. Thus, $(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma'^i) \in [\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n$.
Hence, by assumption, $\mathtt{check\_const}(\{(c_i^{i,i} \cup c'' \cup c', \Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma'^i)\})$ succeeds. If we have $\overline{c}'' = fst(\mathtt{step1}_{\Gamma''}(c''))$, then $\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c'}^i = fst(\mathtt{step1}_{\Gamma_i^{i,i} \cup \Gamma'' \cup \Gamma'^i}(c_i^{i,i} \cup c'' \cup c'))$.
Thus, $\mathtt{step3}_{\overline{\Gamma}_i^{i,i} \cup \overline{\Gamma}'' \cup \overline{\Gamma'}^i}(\overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c'}^i)$ holds, and since $\{M'_1 \sim N'_1, M'_2 \sim N'_2\} \subseteq \overline{c}_i^{i,i} \cup \overline{c}'' \cup \overline{c'}^i$, we know that $N'_1\alpha' = N'_2\alpha'$.

Thus $N_1'\alpha'\delta_i^1 = N_2'\alpha'\delta_i^1$, *i.e.*, since $\delta_i^1\alpha'\delta_i^1 = \alpha$, $N_1\alpha = N_2\alpha$.

Finally, only the case where $M_1 \sim N_1 \in \overline{c}'$ and $M_2 \sim N_2 \in \overline{c}'$ remains. By assumption, $[\ C\ ]_1^n$ and $[\ C\ ]_2^n$ are not empty, hence there exist $(c'', \Gamma'') \in [\ C\ ]_1^n$ and $(c''', \Gamma''') \in [\ C\ ]_2^n$. Thus, $(c'' \cup c''' \cup c', \Gamma'' \cup \Gamma''' \cup \Gamma') \in [\ C\ ]_1^n \cup_\times [\ C\ ]_2^n \cup_\times [\ C'\ ]_1^n$.
Hence, by assumption, $\texttt{check\_const}(\{(c'' \cup c''' \cup c', \Gamma'' \cup \Gamma''' \cup \Gamma')\})$ succeeds. If $\overline{c}'' = fst(\texttt{step1}_{\Gamma''}(c''))$ and $\overline{c}''' = fst(\texttt{step1}_{\Gamma'''}(c'''))$, then $\overline{c}'' \cup \overline{c}''' \cup \overline{c}' = fst(\texttt{step1}_{\Gamma'' \cup \Gamma''' \cup \Gamma'}(c'' \cup c''' \cup c'))$.
Thus, $\texttt{step3}_{\overline{\Gamma}'' \cup \overline{\Gamma}''' \cup \overline{\Gamma}'}(\overline{c}'' \cup \overline{c}''' \cup \overline{c}')$ holds, and since $\{M_1 \sim N_1, M_2 \sim N_2\} \subseteq \overline{c}'' \cup \overline{c}''' \cup \overline{c}'$, we know that $N_1\alpha = N_2\alpha$.
Therefore the claim holds in this case, which concludes the proof that $\texttt{step3}_{\overline{\Gamma}}(\overline{c})$ holds.

Therefore, for every $(c, \Gamma) \in (\cup_{\times 1 \leq i \leq n} [\ C\ ]_i^n) \cup_\times [\ C'\ ]_1^n$, $\texttt{check\_const}(\{(c, \Gamma)\})$ succeeds, which proves the claim.

Lemma 44. *For all (c, Γ) such that $\mathrm{vars}(c) \subseteq \mathrm{dom}(\Gamma)$ which only contains variables indexed by 1 or 2, and all names and keys in c have finite types, if $\texttt{check\_const}(\{(c, \Gamma)\})$ succeeds, then for all $\Gamma'' \in \mathrm{branches}(\Gamma')$, where*

$$\Gamma' = \Gamma[\bigvee_{1 \leq i \leq n} [\![\tau_{m_i}^{l,1}\ ;\ \tau_{p_i}^{l',1}]\!] \ /\ [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]]_{m,p \in \mathcal{N}}[\mathrm{seskey}^{l,1}(T)\ /\ \mathrm{seskey}^{l,\infty}(T)],$$

$\texttt{check\_const}(\{(c, \Gamma'')\})$ *succeeds.*

Proof. Let $n \in \mathbb{N}$.
Let (c, Γ) be as defined in the statement of the lemma.
Let us assume that $\texttt{check\_const}(\{(c, \Gamma)\})$ succeeds. Let

$$\Gamma' = \Gamma[\bigvee_{1 \leq i \leq n} [\![\tau_{m_i}^{l,1}\ ;\ \tau_{p_i}^{l',1}]\!] \ /\ [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]]_{m,p \in \mathcal{N}}[\mathrm{seskey}^{l,1}(T)\ /\ \mathrm{seskey}^{l,\infty}(T)],$$

and let $\Gamma'' \in \mathrm{branches}(\Gamma')$.

The procedure $\texttt{check\_const}(\{(c, \Gamma'')\})$ is as follows:

1. We compute $(\overline{c}, \overline{\Gamma}'') = \texttt{step1}_{\Gamma''}(c)$. Following the notations in the procedure, we denote

$$F = \{x \in \mathrm{dom}(\Gamma'') \mid \exists m, p, l, l'.\ \Gamma''(x) = [\![\tau_m^{l,1}\ ;\ \tau_p^{l',1}]\!]\}.$$

Let $F' = \{x \in \mathrm{dom}(\Gamma) \mid \exists m, p, l, l'.\ \Gamma(x) = [\![\tau_m^{l,1}\ ;\ \tau_p^{l',1}]\!]\}$.
Let also $F'' = \{x \in \mathrm{dom}(\Gamma) \mid \exists m, p, l, l'.\ \Gamma(x) = [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]\}$.
It is easily seen from the definition of Γ' that $F = F' \uplus F''$.
By definition of $\texttt{step1}_{\Gamma''}(c)$, $\overline{\Gamma}''$ contains $\Gamma''|_{\mathrm{dom}(\Gamma'') \setminus F}$.
Let $(\overline{c}', \overline{\Gamma}) = \texttt{step1}_{\Gamma}(c)$.
It is clear from the definitions of Γ' and Γ'' that for all $x \in F''$, there exists $i \in [\![1, n]\!]$ and m, p, l, l' such that $\Gamma(x) = [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]$ and $\Gamma''(x) = [\![\tau_{m_i}^{l,1}\ ;\ \tau_{p_i}^{l',1}]\!]$. Let σ_ℓ and σ_r be the substitutions defined by

$$\mathrm{dom}(\sigma_\ell) = \mathrm{dom}(\sigma_r) = F''$$

and

$$\forall x \in F''.\forall m,p \in \mathcal{N}.\ \forall l,l'.\ \forall i \in [\![1,n]\!].\ \Gamma''(x) = [\![\tau_{m_i}^{l,1}\ ;\ \tau_{p_i}^{l',1}]\!] \Rightarrow (\sigma_\ell(x) = m_i\ \wedge\ \sigma_r(x) = p_i).$$

It is clear from the definition of $\overline{c}$ and $\overline{c}'$ that $\overline{c} = [\![\overline{c}']\!]_{\sigma_\ell,\sigma_r}$.

2. We check that $\texttt{step2}_{\overline{\Gamma}''}(\overline{c})$ holds.
Let $u \sim v \in \overline{c}$. Since $\overline{c} = [\![\overline{c}']\!]_{\sigma_\ell,\sigma_r}$, there exists $u' \sim v' \in \overline{c}'$ such that $u = u'\sigma_\ell$ and $v = v'\sigma_r$.
Since $\texttt{check\_const}(\{(c,\Gamma)\}) = \texttt{true}$, we know that u' and v' have the required form. Note that by definition of $\overline{\Gamma}''$, the keys which are secret in $\overline{\Gamma}''$, *i.e.* the keys $k \in \mathcal{K}$ such that there exist k', T such that $\overline{\Gamma}''(k,k') <: \text{key}^{\texttt{HH}}(T)$ or $\overline{\Gamma}''(k',k) <: \text{key}^{\texttt{HH}}(T)$, are exactly the keys which are secret in $\overline{\Gamma}$. Similarly, for all variable x, there exists T such that $\overline{\Gamma}(x) <: \text{key}^{\texttt{HH}}(T)$ if and only if there exists T such that $\overline{\Gamma}''(x) <: \text{key}^{\texttt{HH}}(T)$; and there exist m, n, a such that $\overline{\Gamma}(x) = [\![\tau_m^{\texttt{HH},a}\ ;\ \tau_n^{\texttt{HH},a}]\!]$ if and only if there exist m, n, a such that $\overline{\Gamma}''(x) = [\![\tau_m^{\texttt{HH},a}\ ;\ \tau_n^{\texttt{HH},a}]\!]$.
It clearly follows, by examining all cases for u' and v', that $u'\sigma_\ell$ and $v'\sigma_r$, *i.e.* u and v, also have the required form.
Therefore, $\texttt{step2}_{\overline{\Gamma}''}(\overline{c})$ holds.

3. Finally, we check the condition $\texttt{step3}_{\overline{\Gamma}''}(\overline{c})$.
Let $M_1 \sim N_1 \in \overline{c}$ and $M_2 \sim N_2 \in \overline{c}$. Since $\overline{c} = [\![\overline{c}']\!]_{\sigma_\ell,\sigma_r}$, there exist $M_1' \sim N_1' \in \overline{c}'$ and $M_2' \sim N_2' \in \overline{c}'$ such that $M_1 = M_1'\sigma_\ell$, $N_1 = N_1'\sigma_r$, $M_2 = M_2'\sigma_\ell$, and $N_2 = N_2'\sigma_r$.
Let us prove the first direction of the equivalence, *i.e.* the case where M_1, M_2 are unifiable. The proof for the case where N_1, N_2 are unifiable is similar.
If M_1, M_2 are unifiable, let μ be their most general unifier. We have $M_1\mu = M_2\mu$, *i.e.* $(M_1'\sigma_\ell)\mu = (M_2'\sigma_\ell)\mu$.
Let τ denote the substitution $\sigma_\ell\mu$. Since $M_1'\tau = M_2'\tau$, M_1' and M_2' are unifiable. Let μ' be their most general unifier. There exists θ such that $\tau = \mu'\theta$.
Let also α be the restriction of μ to $\{x \in \text{vars}(M_1) \cup \text{vars}(M_2) \mid \overline{\Gamma}''(x) = \texttt{LL}\ \wedge\ \mu(x) \in \mathcal{N}\}$.
Note that $\overline{\Gamma}''(x) = \texttt{LL} \Leftrightarrow \overline{\Gamma}(x) = \texttt{LL}$.
We have to prove that $N_1\alpha = N_2\alpha$.

Let $x \in \text{vars}(M_1') \cup \text{vars}(M_2')$ such that there exist m,p,l,l' such that $\overline{\Gamma}(x) = [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]$, *i.e.* $x \in F''$.
By definition of σ_ℓ (point 1), there exists $i \in [\![1,n]\!]$ such that $x\sigma_\ell = m_i$ (and $x\sigma_r = p_i$). Hence, we have

$$(x\mu')\theta = x\tau = x\sigma_\ell\mu = (x\sigma_\ell)\mu = m_i\mu = m_i.$$

Thus, $x\mu'$ can only be either a variable y such that $y\theta = m_i$, or the nonce m_i.
Therefore, μ' satisfies the conditions on the most general unifier expressed in $\texttt{step3}_{\overline{\Gamma}}(\overline{c}')$.

Let $x \in \text{vars}(M_1) \cup \text{vars}(M_2)$ such that $\overline{\Gamma}''(x) = \texttt{LL}$ and $\mu(x) \in \mathcal{N}$. We have $(x\mu')\theta = x\tau = x\sigma_\ell\mu = (x\sigma_\ell)\mu = x\mu = \mu(x) \in \mathcal{N}$. Thus, $x\mu'$ can only be either a variable y (such that $y\theta = \mu(x)$), or the nonce $\mu(x)$.
Conversely, let $x \in \text{vars}(M_1') \cup \text{vars}(M_2')$ such that $\overline{\Gamma}(x) = \texttt{LL}$ and $\mu'(x) \in \mathcal{N}$. We have $x\mu = (x\sigma_\ell)\mu = x\tau = (x\mu')\theta = \mu'(x)$.

Let then θ' be the substitution with domain $\{x \in \text{vars}(M_1') \cup \text{vars}(M_2') \mid \exists m,p,l,l'.\exists i \in [\![1,n]\!].\ \overline{\Gamma}(x) = [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]\ \wedge\ \mu'(x) = m_i\}$ such that $\forall x \in \text{dom}(\theta').\ \theta'(x) = p_i$ if $\mu'(x) = m_i$ and $\overline{\Gamma}(x) = [\![\tau_m^{l,\infty}\ ;\ \tau_p^{l',\infty}]\!]$.
Let also α' be the restriction of μ' to $\{x \in \text{vars}(M_1') \cup \text{vars}(M_2') \mid \overline{\Gamma}(x) = \texttt{LL}\ \wedge\ \mu'(x) \in \mathcal{N}\}$.

Since $\texttt{check\_const}(\{(c, \Gamma)\}) = \texttt{true}$, we know that $\texttt{step3}_{\overline{\Gamma}}(\overline{c}')$ holds. Since $M_1' \sim N_1' \in \overline{c}'$, and $M_2' \sim N_2' \in \overline{c}'$, this implies that $N_1'\alpha'\theta' = N_2'\alpha'\theta'$.

As we have just shown, for all $x \in \mathrm{dom}(\theta')$, there exists $i \in [\![1, n]\!]$ such that $x\sigma_\ell = m_i$ and $x\sigma_r = p_i$, and $\mu'(x)$ is either m_i or a variable. By definition of $\mathrm{dom}(\theta')$, only the case where $\mu'(x) = m_i$ is actually possible, and we have $\theta'(x) = p_i$.
Thus, $\forall x \in \mathrm{dom}(\theta').\ \sigma_r(x) = \theta'(x)$.
It then is clear from the definitions of the domains of θ' and σ_r that there exists τ' such that $\sigma_r = \theta'\tau'$.

Thus, since we have shown that $N_1'\alpha'\theta' = N_2'\alpha'\theta'$, we have $(N_1'\alpha'\theta')\tau' = (N_2'\alpha'\theta')\tau'$, that is to say $N_1'\alpha'\sigma_r = N_2'\alpha'\sigma_r$, *i.e.*, since α' and σ_r have disjoint domains, and are both ground, $N_1\alpha' = N_2\alpha'$.

Moreover, we have shown that for all $x \in \mathrm{vars}(M_1') \cup \mathrm{vars}(M_2')$ such that $\overline{\Gamma}(x) = \texttt{LL}$ and $\mu'(x) \in \mathcal{N}$, $\mu(x) = \mu'(x)$. That is to say that for all $x \in \mathrm{dom}(\alpha')$, $\mu(x) = \alpha'(x)$.
In addition, it is clear from the definition of σ_ℓ that

$$\{x \in \mathrm{vars}(M_1') \cup \mathrm{vars}(M_2') \mid \overline{\Gamma}(x) = \texttt{LL}\} = \{x \in \mathrm{vars}(M_1) \cup \mathrm{vars}(M_2) \mid \overline{\Gamma}''(x) = \texttt{LL}\}.$$

Hence

$$\begin{aligned}
\mathrm{dom}(\alpha) &= \{x \in \mathrm{vars}(M_1) \cup \mathrm{vars}(M_2) \mid \overline{\Gamma}''(x) = \texttt{LL} \ \wedge\ \mu(x) \in \mathcal{N}\} \\
&= \{x \in \mathrm{vars}(M_1') \cup \mathrm{vars}(M_2') \mid \overline{\Gamma}(x) = \texttt{LL} \ \wedge\ \mu(x) \in \mathcal{N}\} \\
&\supseteq \{x \in \mathrm{vars}(M_1') \cup \mathrm{vars}(M_2') \mid \overline{\Gamma}(x) = \texttt{LL} \ \wedge\ \mu(x) \in \mathcal{N} \ \wedge\ \mu'(x) \in \mathcal{N}\} \\
&= \{x \in \mathrm{vars}(M_1') \cup \mathrm{vars}(M_2') \mid \overline{\Gamma}(x) = \texttt{LL} \ \wedge\ \mu'(x) \in \mathcal{N}\} \\
&= \mathrm{dom}(\alpha').
\end{aligned}$$

Therefore, $\forall x \in \mathrm{dom}(\alpha').\ x \in \mathrm{dom}(\alpha) \ \wedge\ \alpha'(x) = \alpha(x)$. Thus there exists α'' such that $\alpha = \alpha'\alpha''$.

Since we already have $N_1\alpha' = N_2\alpha'$, this implies that $N_1\alpha = N_2\alpha$, which concludes the proof that $\texttt{step3}_{\overline{\Gamma}''}(\overline{c})$ holds. Hence, $\texttt{check\_const}(\{(c, \Gamma'')\}) = \texttt{true}$.

We can now prove the following theorem, which corresponds to the second step necessary for Theorem 3, mentioned in Subsection 6.4:

Theorem 11. *Let C, and C' be two constraint sets without any common variable*

$$\texttt{check\_const}([\,C\,]_1 \cup_\times [\,C\,]_2 \cup_\times [\,C'\,]_1) = \texttt{true} \ \Rightarrow \forall n.\ [\,C'\,]_1^n \cup_\times (\cup_{\times 1 \le i \le n} [\,C\,]_i^n) \textit{ is consistent.}$$

Proof. Assume $\texttt{check\_const}([\,C\,]_1 \cup_\times [\,C\,]_2 \cup_\times [\,C'\,]_1) = \texttt{true}$. Let $n > 0$.
Let us show that $[\,C'\,]_1^n \cup_\times (\cup_{\times 1 \le i \le n} [\,C\,]_i^n)$ is consistent.
By Theorem 9, it suffices to show that $\texttt{check\_const}([\,C'\,]_1^n \cup_\times (\cup_{\times 1 \le i \le n} [\,C\,]_i^n)) = \texttt{true}$.
By Theorem 10, it suffices to show that $\texttt{check\_const}([\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n) = \texttt{true}$.
By assumption, we know that $\texttt{check\_const}([\,C\,]_1 \cup_\times [\,C\,]_2 \cup_\times [\,C'\,]_1) = \texttt{true}$.
That is to say, for each $(c_1, \Gamma_1) \in C$, $(c_2, \Gamma_2) \in C$, $(c_3, \Gamma_3) \in C'$, if $c' = [\,c_1\,]_1^{\Gamma_1} \cup [\,c_2\,]_2^{\Gamma_2} \cup [\,c_3\,]_1^{\Gamma_3}$, and $\Gamma' = [\,\Gamma_1\,]_1 \cup [\,\Gamma_2\,]_2 \cup [\,\Gamma_3\,]_1$, $\texttt{check\_const}(\{(c', \Gamma')\}) = \texttt{true}$.
Thus, by Lemma 44, for all $(c_1, \Gamma_1) \in C$, $(c_2, \Gamma_2) \in C$, $(c_3, \Gamma_3) \in C'$, if $c' = [\,c_1\,]_1^{\Gamma_1} \cup [\,c_2\,]_2^{\Gamma_2} \cup [\,c_3\,]_1^{\Gamma_3}$, and $\Gamma' = [\,\Gamma_1\,]_1^n \cup [\,\Gamma_2\,]_2^n \cup [\,\Gamma_3\,]_1^n$, $\texttt{check\_const}(\{(c', \Gamma')\}) = \texttt{true}$.
That is to say, $\texttt{check\_const}([\,C\,]_1^n \cup_\times [\,C\,]_2^n \cup_\times [\,C'\,]_1^n) = \texttt{true}$, which concludes the proof.